SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	State of the Art in Mashup tools
Document version:	**V-0.25**
Task code:	
Deliverable code:	D4.2.1
WP Leader (organisation):	OrangeLabs (FT)
Deliverable Leader (organisation):	OrangeLabs
Authors (organisations):	OrangeLabs, EBM, EMAC
Date of first version:	January 2011

Table of Contents

1. Executive Summary

The goal of SocEDA is to develop and validate an elastic and reliable federated SOA architecture for dynamic and complex event-driven interaction in large highly distributed and heterogeneous service systems. Such architecture will enable exchange of contextual information between heterogeneous services, providing the possibilities to optimize/personalize the execution of them, according to social network information. The main outcome will be a platform for event-driven interaction between services, that scales at the Internet level based on the proposed architecture and that addresses Quality of Service (QoS) requirements. The platform consists of:

1. Federated middleware layer: a peer-to-peer overlay network combined with a publish/subscribe mechanism, that has the task to collect events coming from the heterogeneous and distributed services
2. Distributed complex event processor: an elastic, distributed computing cloud based engine for complex processing of events coming from different services in order to detect interesting situations a service should react on
3. Social aware event modeling and matching plus an event based workflow engine for filtering events and proposing adaptation and changes in running business processes and services
4. A monitoring and governance framework as well as a Mashup frontend to describe and manage services as well as business events.

This document focuses on the 4th point and more particularly on Mashup tooling, taking into account the Mashup designer point of view. It gives an overview of the mashup domain, its evolution since it appeared as part of the Web 2.0 technologies, and it explains the positioning we chose for a mashup designer, regarding its possible capabilities and our particular needs.

During this study, we selected representative Mashup tools for a detailed evaluation, from graphical components composers to process designers. Looking for their main interesting features in the context of SocEDA, the question of mashup functional and business perimeter is being examined, pushing the review to BPM tools (i.e. process orchestrators), which are also composers. Key points regarding inclusion of events in compositions, as well as simplicity allowing non-developer user manipulations is being especially considered.

Finally, after theoretical and practical mashup field has been explored, this State of The Art is describing the expected requirements for a Mashup tool in the context of the SocEDA platform.

2. Introduction

2.1. Objectives

SocEDA project aim is to provide an open distributed platform for event-driven interaction between services that scales at the Internet level. To achieve this goal, a federated architecture will be deployed to address the multiplicity and the heterogeneity of service networks. On top of this platform, a tool will be offered to business users to define complex event patterns and link them with business processes.
In order to allow business services to be designed by non developer users[1], we may provide a graphical tool which will address this need in a user-friendly manner.
In order to introduce events in a business process design, we may provide a powerful software able to design services orchestration, through a large panel of operations, including in particular the attempt of an event and the interruption by an event occurring.
Mashup design ought to be designated to achieve those goals. Arose to “mash” Internet data and services, most of them offer graphical interface allowing end-user to easily create their enriched services. Today, this technology has a few years of existence and it begins to offer the required sophistication to sequence services finely and compose business processes.
Mashup designer tools are numerous and they address very different needs. So a large overview of the available software of the market, as a deep analyze of at least a subset of them is necessary to evaluate what can be achieved with such a tool in SocEDA.
In this perspective, this document will list available software achieving mashup, and report a selective evaluation about a subset of them, in order to:

- picture the overall level of development of graphical mashup solutions, the functionalities offered and their maturity, the tools available for non developer users,
- frame and structure, through selective criteria, the evaluation of the listed software,
- refine requirements for the design tool that is expected in SocEDA platform,
- determine if one of evaluated software could be embedded in the platform as it is,
- determine the work to enhance it to fulfill the whole platform needs, or if the expected solution should be implemented.

2.2. Mashup and mashup design tools

Mashups are the result of integration of contents, logic and interfaces available on the Internet. They are contributing to information sharing and interoperability through a user-centered design. So they can be considered as one of the technologies characterizing the Web 2.0[8].
In this section we’ll give the main explanation to understand what a mashup is, and how it can be designed and executed. Then, we’ll point to its capabilities and limitations, given its own nature but considering also mashup design tools power.

[1] or “Business users”, so called in SocEDA project Objectives

2.2.1. Context and definitions

The definition of mashup slightly varies from literature sources, as well as its composition capabilities vary from editor's solutions. For some, mashup is defined as a data aggregation, combining presentation to content, through an end user Web page, as a portal. Others see mashup overlapping the business process field, designed as a services orchestration resulting in rich business application. As an example of the first vision, the OpenAjax Alliance considers a mashup as *a Website or application that combines content from more than one source into an integrated experience* [1].

Many other sources [2] agree, as Gartner:

A mashup is a way to build composite applications that possess three fundamental characteristics:

- *They are lightweight, and they employ the technologies and principles of the Web.*
- *They source content or functionality from existing systems, and they have no native data store or content repository.*
- *Their result is an explicit mixture of source content and functionality, where the sourced content and functionality retain their original essence.*

Mashups were first used on social and consumer Web, taking advantage of Web 2.0 technologies: Asynchronous JavaScript and XML (AJAX), Really Simple Syndication (RSS), Atom feeds, Web Services and Web pages.
The first known mashup began with the application programming interface (API) provided by Google for Google Maps and then Google Earth [3]. Google but also Amazon, eBay and others, made available thousands of APIs transforming the Web in a large platform of collaborative development and information sharing. Knowing the success it encountered, and business benefits it may generate, it's easy to understand that editors and major software players took interest in this business. Indeed, mashups imply no complex coding, are quick to adapt to frequent changes, and would be able to address the "long-tail" of users needs. Environments to build and deploy mashups multiplied then, driving to more sophistication, security and access control enhancement. Also, the scope of mashup composition widen, introducing the process in the theoretical capabilities of mashup assembly.
So the mashup definition evolve, and one can read in the literature "SOA orchestration" so called "Process Mashup" [4]. As well, editors offer to "draw workflow" with mashup composer. Cordys promotes "process-centric" mashup, and Convertigo claims its Enterprise Mashup Server allows "business process orchestration". In this study, we will retain the ability to sequence processes in the mashup perimeter, and will define three types of composition addressable by mashup tools:

- Presentation mashup where composition is made at user interface level. Interactions between data from different sources and services calls can be configured,
- Data mashup, where data is retrieved from different sources and systems. Tools offering this type of composition have to include connectors for a wide range of existing data sources as well as mash mechanisms like "join", "filter",
- Process workflow where different Web Services are called and orchestrated to create a composite application. Resulting application can be either exposed as a web Service, if back-end processes are being defined, or proposed as a stand alone application.

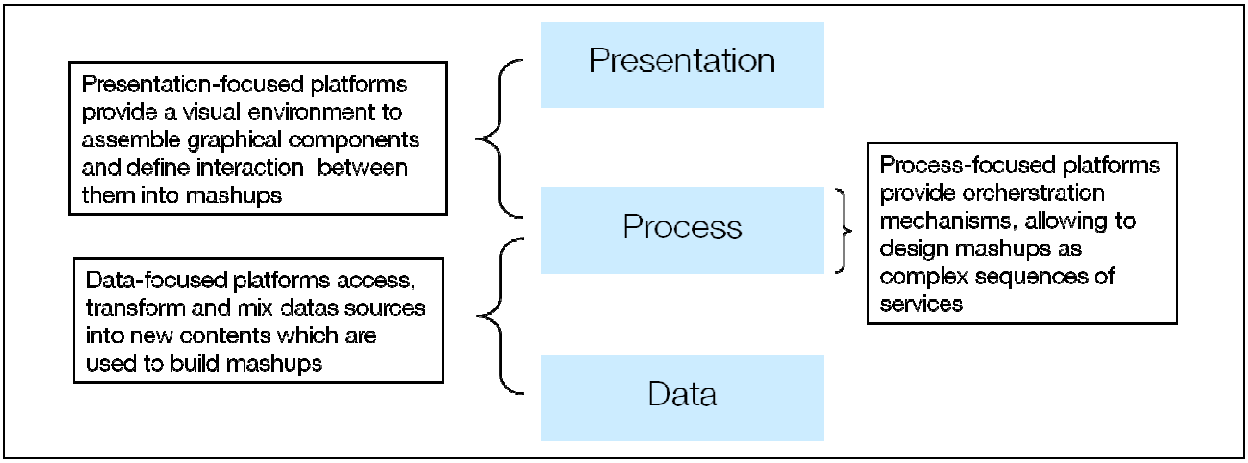

Figure 1 : mashup segmentation

Other mashup segmentation exists, categorizing between *consumer* versus *developer* mashups, whether a graphical design tool is used or a language. Often *enterprise* mashups and *business* mashups are differentiated, whether the content source is partially or totally inside the enterprise, and whether the resulting application is destined to end user or business users. Also, *data* mashups can be considered as a new composite source content spared of any presentation component. In this document, we will consider that enterprise/business mashup could cover the three level of the segmentation above, and still expect mashup design tools to be intuitive and open to non developer users.

2.2.2. Technical principles

The figure below categorizes the technologies involved in mashup development [5]:

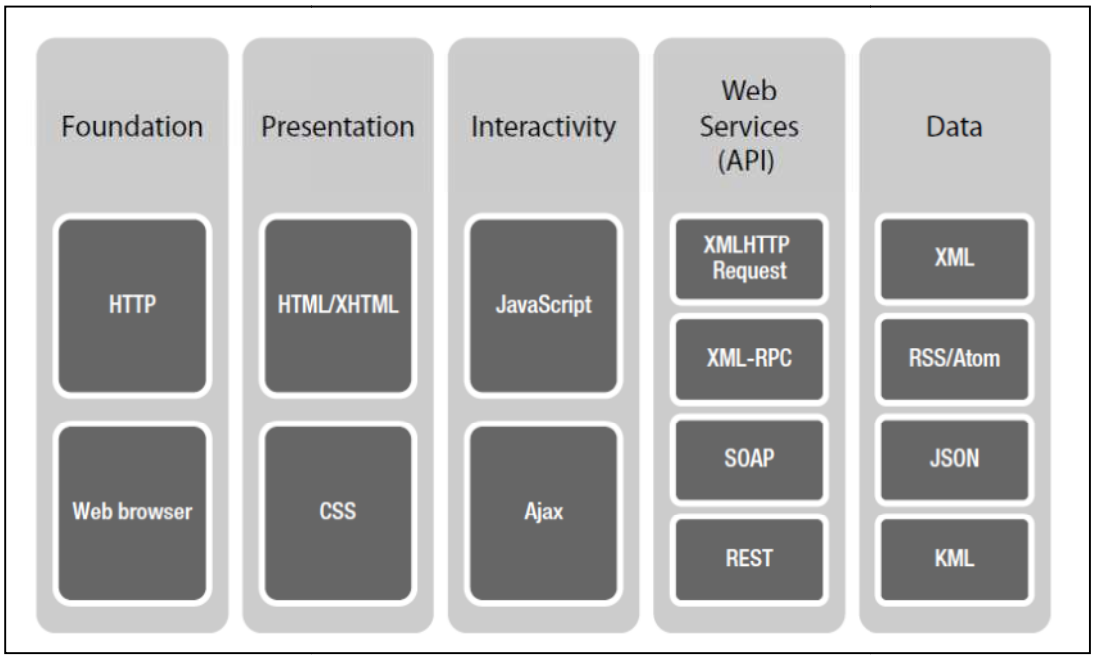

Figure 2 : mashup based-on technologies

At runtime, composite application graphical components (may be called[2] widget, gadget, mashlet) are accessible in a Web browser. They interact with APIs and between each other through JavaScript mechanisms and access content encapsulated in a text-based Web standard.

At design time in a user-friendly tool, there would be business components available in a palette for drag and drop. They would be parameterized through text boxes (like APIs URL setting) and interactions between business components would be materialized by arrows. The tool would generate simple XML description for the resulting composite application.

A very simple example of such graphical design would look like this[3] :

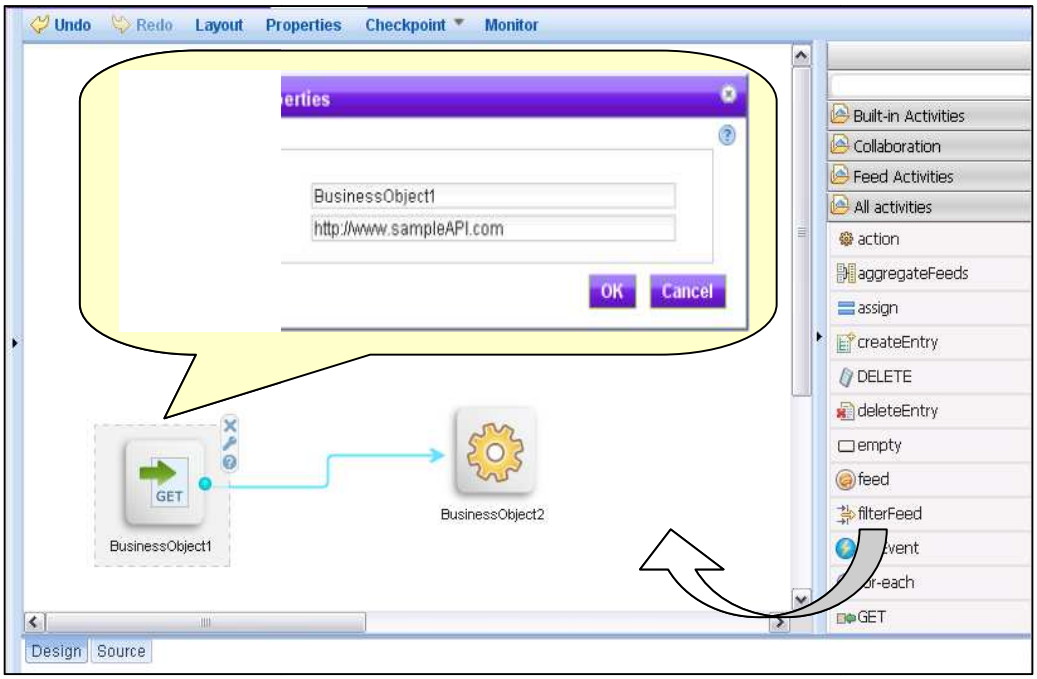

Figure 3: example of basic composition, graphical view

The resulting textual definition would look something like this[4]:

```
<mashup          name="MyTestMashup"  xsi:schemaLocation= ... >
  <operation name="runMashup" >
        <URLInvocation          name="BusinessObject1"
                                Url="http://www.sampleAPI.com"
                                method="receiveGET
                                output="BO1_out" />
        <ReceiveAndDisplay      name="BusinessObject2"
                                input="BO1_out"
                                method="display" />
   </operation>
</mashup>
```

Figure 4: example of basic composition, textual view

[2] Classically called "Widgets", graphical components displayable in a Web Brower are called "Gadget" by Google, “Mashlet” by Presto 2.7. The different names used may imply nuance, no real difference.
[3] This view has been created in IBM sMash environment.
[4] This XML description is inspired from EMML (OMA) language.

2.2.3.Mashup positioning, capabilities and limitations

About mashup positioning, one can read the following assertions:

- Mashups composition extends the concepts of Service-Oriented Architecture (SOA) with the Web2.0, offering in addition user interface integration [6],
- Mashups sources extend the Software as a Service (SaaS) offer to APIs distributed in different platforms across the internet [7].

Referring to the above statements and given the wide mashup definition we have chosen, it appears we read marketing speech. One could believe any type of application can be mashed-up, by a non developer user, responding to dedicated needs and at the less cost. This is of course feasible, as far as the necessary APIs and graphical tool capabilities are available. Though, even all necessary APIs and the most powerful graphical tool are available, mashup limits are large and various. Through the above list of subject, we'll try to browse mashups limits and capabilities, and see how mashup positioning is influenced by its characteristics.

APIs availability

The first limit to mention is the APIs availability, and indeed, if they're numerous on the Web, they do not cover every need. Thus, to compose a particular mashup, one may have to include a part of development, or wait for the APIs offer to broaden. Moreover, to appear as a safe choice for business products, APIs must ensure compatibility through future versions, which might be not offered by APIs providers. Though, those two problematic exist as well on the SaaS market.
The trend today is clearly to API openness, and the increasing evolvement of major actors as well as individuals, on those API development and use, suggests the offer will keep growing and enhancing. Though, no one can say if that bubble would not burst.

Perimeter

The second limit to mention is the capabilities of a design tool to cover the three composition levels defined (Presentation/Data/Process), and still offers an intuitive way to compose complex applications. This question has been detailed in this document's introduction, and will be addressed through the tools review in following chapters.

Standardization

One limitation in the mashup world is the current lack of standard, restricting the portability and interoperability. As we included the process orchestration in the mashup perimeter, we should mention two well known and improved standards that contradict the previous assumption: BPMN for design and BPEL for execution[5]. Though, if we restrict mashup perimeter to its primary definition, we may search for a more data-mashing oriented language, if we expect composition to be accessible to non-developer users, we may look for simpler composition notation.
An initiative of this kind has been launched by JackBe [11]. The result is the creation of a consortium, the Open Mashup Alliance (OMA)[17], regrouping actors from enterprise mashup (as DreamFace[10] , Convertigo[18] , Kapow[19]) and from SOA and Web 2.0 worlds. JackBe opened its mashup language to

[5] We're referring to the "Process" subset of those two standard, See newt section (Mashup versus Business Process Management) for details about Mashup related to BPM

create a reference mashup language, the *Enterprise Mashup Markup Language* (EMML). But works still have to be done to have EMML introduced in standard body like OASIS or World Wide Web Consortium. Below, the OMA proposal for mashup standardization through EMML:

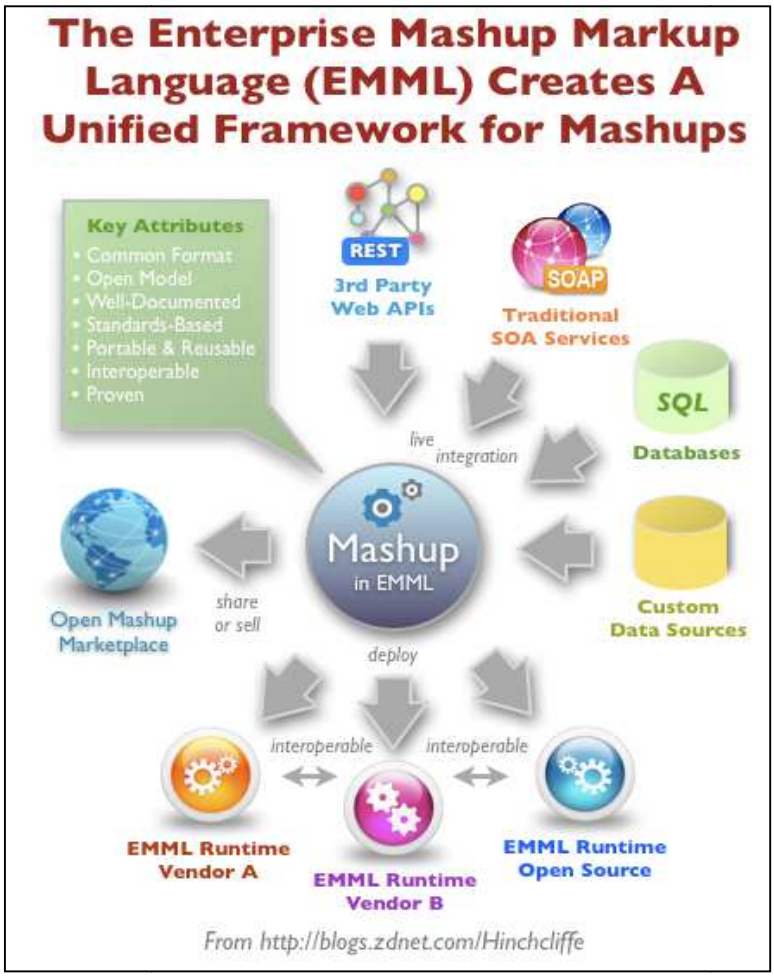

Figure 5 : mashup standardization through EMML

Security

Another important limit concerns the security. Because mashups are dynamic and multi-domain by nature, they introduce *new threats to the already-existing insecure landscape*[3]. Indeed, during mashup execution, client-side scripts exchange data with any remote Web servers, which data is authorized to pass through firewall and reach potential back-end systems, with limited human oversight. This may easily introduce malicious contents in systems. For example, a script embedded in an image attribute, as it's executed in the client browser, can allow information to be stolen, as password or cookies information. Another example is a SQL hacking request that could be introduced instead of a legal request during the mashing of content.

To prevent from these risks, application logic, components interfaces and possible user actions should be considered and monitored. Though, big efforts have to be done to offer little security guarantee. For this reason, in the same but reinforced principle than the "same-origin policy" (SOP) browser control,

preventing HTML scripts loaded in one origin to access document from an other, a complementary approach would be to develop and promote trust-labels mechanisms. This will allow certifying APIs, components or composite applications origin and innocuousness.

Quality – Reliability

Like the security question, this field is quite important to address in the perspective of mashup expanding, i.e. components delivery growth and tools development and enhancement. In 2009[8], 62% of CIOs were seeing mashup as not as reliable as traditional applications and not adapted for enterprises [8].
Mashup reliability, as mashup quality, depends both on each component and on orchestration mechanisms. This second point is the responsibility of the mashup design tool, but is also conditioned by components themselves, through the interfaces they offer and data standards they produce and consume.
So beside the component's inner qualities, there's a real challenge to ensure the component's *mashability* [9]. This quoted property mainly includes compatibility of components all together, on a technical level, through languages and data format, and on a syntactic plus semantic level, through input/output interactions.

Mashup versus Business Process Modeling

In order to avoid confusion and to complete mashups positioning, we should relate *business mashups* to business process management. Regarding modeling, they appear applying to the same design principles, and regarding execution, they consequently may be thought quite similar.
Many points differentiate them though, but before to address them, we may first define the intersection space where they can be compared. Indeed, BPM Notation covers a large scope from which mashup composition is excluded. The first one is abstract modeling, where business users describe their activities sequencing uncorrelated with underlying concrete services. The second is choreography modeling, specifying existing services interactions with no binding to a centralized execution. So, the restricted BPM portion where we'll try a comparison with mashup is the one related to Business Process orchestration mechanisms, on its design part through the associated notation, and on its execution part through BPEL.
On the mashup side, we will focus on *process-centric* mashups, including or not other composition levels.
In this area, *process-centric* Mashup and BPM designs inspire comparison because they look alike, even though the second is based on a far richer language. Actually, they are separated by an abyss of sophistication, BPMN intrinsic, including: actor's role definitions, monitoring and auditing properties, elaborated control structures. But nothing would prevent mashup designers to implement such a complexity, until they can lean upon remote services fulfilling their associated needs. Well, it may be where we touch the real difference between mashups and BPM, through underlying services "dedicated" to mashups, and because of the typology of mashup users.
Referring to the literature, mashup history and most of the current design tools, we learn that mashups target individual users, that they call internet services (basic on their use as on the guaranties they offers), and that they should be designed in intuitive and user-friendly environments.
On the contrary, BPMN/BPEL is considered as the serious, secure and thus proper manner to model and execute business processes, not only because of its capabilities as standards, but also because it rather leans on inner services, not on services of the Internet.

Though, as the Business Process Management is complicated and hardly progressive, could *process-centric* mashups be the right compromise to address particular business needs, with more simplicity in the design and flexibility in the maintenance? For the particular question that has our interest here, does

events in service composition have to be abandoned to elite designers because it implies a complexity that mashup composition tools currently do not address ? We will try to answer the question in this document, through the tools review. It will show if there are existing solutions where mashup composition fully includes the process level, and if not, if enhancement would be feasible to reach this theoretical border, enough at least to fulfill our requirements.

2.3. Acronyms

Acronym	Definition
CEP	Complex Event Processing
WS	Web Service
SOA	Service Oriented Architecture
WSDL	Web Service Description Language
SLA	Service Level Agreement
EMML	Enterprise Mashup Markup Language
Mashup[6]	Composite applications that combine multiple, disparate data sources and functionalities into something new. Broad layers of item types that can be mashed together: Presentation, Data and Process.
BPM	Business Process Modeling or Business Process Management
BPMN	Business Process Model and Notation
BPEL	Short for Web Services Business Process Execution language (WS-BPEL)

[6] See Chapter 2.2.1 - Context and definitions

3. Methodology

3.1. Approach

We place our context within the framework of an event based service oriented platform for companies.
This platform offers a mashup composition tool, allowing a non developer user to create simple services or more complex service compositions. Simple or enriched events should intervene in the service composition, as trigger to designed services, as interruptions or as a result of their execution.
On SocEDA platform, a CEP editor will allow to design complex event rules producing new enriched events. Yet, complex event rules design is part of the process of designing services, whether it is been achieved prior to user's service design or by the user himself. Thus, both those designs should preferably be realized in an integrated environment.
In order to fulfill our requirements, we are looking for a tool with a user-friendly interface, capable to assemble components graphically and to design handling of incoming events, at process start or during the process flow, as well as capable to product outgoing events. Also, this environment should allow to design or configure simple rules for the triggering of new event interaction with services components.
At last, we'll look for a tool able to establish the link with the complex event edition, in order for mashups to easily integrate new enriched events. Thus, we'll take a particular interest at the APIs offered and at the layer separation and the exchange formats available.
Of course, as the mashup tool is part of the platform, it should offer interfaces with the workflow engine.
As we can't evaluate the complete set of mashup tool in the market, and as we are looking as well for graphical capabilities than complex processing abilities, we chose to first list a large number of tools[7] and selected a subset.

3.2. Mashup tools selection

Our selection has been based on our knowledge of the current tools available, and when we had no practical experience, on literature or editors descriptions and show cases. We took into account different criteria, including software maturity and graphic tooling availability. We focused on tool's composition perimeter, in order to evaluate one tool of each kind. We also retained particular software, as Presto for its involvement in defining a mashup standard language, and WSO2 as it offers an open source complete solution.
Below is the list of tools we considered containing commercial as well as open source products. Of course, the open sources ones will be favored, though we don't reduce our scope to those, as we're looking for powerful functionalities, on a market that is still not mature.

- *DreamFace* : DreamFace is a widget-based framework to build, use, and distribute enterprise Web 2.0 applications and mashups[10]
- *Presto* : Presto defines itself as an Enterprise Mashup and an App Store[11]
- *WSO2 mashup*: WSO2 Mashup Server delivers enterprise-class service composition[12]
- *IBM WebSphere Smash*: WebSphere sMash introduces a simple environment for creating, assembling and running applications based on popular Web technologies[13]
- *Cordys MashApps Composer*: Cordys Process Factory is a Platform as a Service (PaaS) [14]
- *Bonita Open solution*: Bonita Open solution is a very smart and flexible BPMN design tool[15]
- *GEasyBPMN Editor* : Geasy BPMNEditor is an open source, web-based BPMN 2.0 editor [16]

[7] See Chap. 9.1 - Mashup tools base list in 9Appendix section

3.3. Relevant criteria

This detailed comparison will be based on general and specific criteria to assess technical and non technical aspects of our preselected solutions.

3.3.1. Basic aspects

1) User interface

This service and event composition platform should be accessible by users with basic knowledge in computer science. So, we will have to check the facilities offered by the tool, i.e. the user type targeted.

2) Composition Perimeter

A mashup composition tool may combine data, services or functionalities, and/or presentation components to create enriched services. Depending on the composition range, it addresses different needs: workflow management, back office treatments, GUI creation, feeds aggregation.

3) Life cycle

The creation of a composite application is based on different steps, the component creation, component assembly, test, and deployment. We would be interested by a solution embedding the whole life cycle.

4) Roadmap

This criterion evaluates the maturity and evolution planned for the next version. This is also a warranty of a certain activity around the solution for the next years.

5) Support and community

This criterion evaluates the community support. This is relevant in various aspects, such like:

- Taking advantage of a fulfilled component library to leverage the ability of business application creation.
- Wealth of documentation (forums, blog, wiki).

6) Technical documentation

What is the quality of the documentation? How structured, complete is it? Are samples applications offered? This aspect is quite relevant because it conditioned the time necessary to get trained in the use of the tool.

7) License

This important criterion influences the choice. The tool's license, from open source to commercial will be an important criterion.

3.3.2. Technical aspects

8) Functional scope

The tool is checked according to the needs of mashup design, including event processing aspect. The ability to formalize expressions representing complex event rules and the delegation of their evaluation

will also be evaluated. More generally, event management will be studied, as a native functionality, or as a possible extension. To check those aspects the tool will be tested through the scenarios executions.

- Component creation
- Assembly
- Deployment
- Event manager
- Interaction between Event manager and application component

9) Design/Runtime separation

The mashup tool may only be selected for its graphical process design capabilities, or if the runtime layer can be used. We may need to introduce the event specificities between the design composition and its execution. Consequently, this important specificity has to be evaluated.

10) Interoperability: data format, import/export facilities, language

We need to know which export/import formats are available in order to consider the interoperability of the solution. We will also consider supported languages (for example we could be interested by tomorrow's standard such like EMML).

11) Interoperability: Standard based architecture, APIs

Beyond the input/output format aspect, that may limit the study at possible static exchanges, we need to see further how flexible the solution is, and if it can be embedded in a platform (or how dynamically the platform could interoperate with it). We will evaluate if the tool is based on well-known standards such like Web Services.

12) Event APIs

At runtime, complex event treatment, or even simple event treatments, will have to be delegated to a CEP engine. This binding may take several forms: native API toward a specific CEP engine, JMS API, or other kind of connector allowing event treatment delegation.
At design time, the tool should support events description, incoming event handling and outgoing event production representations. Those descriptions should be concretized at runtime through binding to Event APIs.

13) Possible enrichment and configuration

We are interested here in the ability of the tool to support new components, designed with the tool itself or imported through a standard format (here input format is to be understood as describing components, not data sources).
If it's the case, we'll have to know how easily it can be done, how far new components can be reused, if they are configurable and if they can be stored in a structured way.

4. Mashup Design Tools

4.1. DreamFace

DreamFace is a widget-based framework to build, use, and distribute enterprise Web 2.0 applications and mashups. It is an open source Ajax framework mainly developer oriented, it allows to create and share widgets through a method called Web Channels.

4.1.1. User interface

The GUI is simple and rather accessible to a non developer, though it is to do nothing too complex.
Even with the beta version (V3) which offers a widget editor, some JavaScript should be written to create widget with new functionalities.

4.1.2. Composition perimeter

DreamFace allow data combining and mapping, and offers widget creation tools to present the result.
It allows assembling created widgets, as an interacting set, and then creates rich Web pages.
As a result of composition, widgets react on user events.
No task sequences can be set through composition.

4.1.3. Life cycle

Creation of widgets, assembly and deployment are managed through the tool. There's no test environment, the results can be evaluated through graphical manipulations on deployed widgets.

4.1.4. Roadmap

The site is not on date, but the first months 2011 should be dedicated to debug V3 version and produce a proper documentation.

4.1.5. Support and Community

A very light activity of the forum community

4.1.6. Technical documentation

No user documentation is provided for design operations. A light JavaScript description API is available.

4.1.7. License

DreamFace 2.0 is available under a dual-license open source model that is based on GPL 3.0.

4.1.8. Functional scope

A component in DreamFace should be understood as a widget. DreamFace allow creating new widgets and assembling widgets all together. It provides mechanisms to carry out data interactions between those widgets. The tool manages HTML deployment, and provides, when an application has been created, the URL to access it. DreamFace implements no principle of event management.

4.1.9.Design/runtime separation

The DreamFace's runtime is Tomcat, under which the tool is deployed itself. Though, both the tool and the applications produced are embedded.

4.1.10. Data format, export/import facilities, language

It uses XML compliant DataSource (RSS, SOAP and XML). DreamFace V3 Beta allows exporting mashups to import in another DreamFace environment, but the V2.0 does not.

4.1.11. Standard based architecture, APIs

DreamFace can invoke SOAP and REST Web Services. DataSources can read JSon and XML files.
Connectors are available for databases and files, or can be developed in Java.
A JavaScript API is provided. No API exists to access DreamFace's functionalities.

4.1.12. Event APIs

Event concept in DreamFace is limited to the user interaction with the GUI.

4.1.13. Possible enrichment and configuration

New Widget can be created automatically, based on data sources for the fields, and customizable for appearance. Widget with customized functionalities can also be written in JavaScript (the V3 - Beta offers a Widget editor to enhance the widget customization). Available widgets can be browsed through a list and they can be filtered by category.

4.2. Presto (JackBe)

Presto defines itself as an Enterprise Mashup and an App Store. Presto is also the creator of EMML, a mashup composition language that aims to become a standard. Presto initiate the OMA consortium, in association with a dozen of partners, in order to promote mashup technologies and the adoption of EMML.
Two Presto versions have been evaluated: the stand alone version 2.7 and the 3.0 cloud version.
There are no fundamental changes between both, but a larger number of EMML functionalities became available for graphical manipulations.
Presto offers an environment to create mashups, graphically or using EMML, associate them to Web views, customize the designed components through complete Web applications, and make those applications available on the Presto App Store.

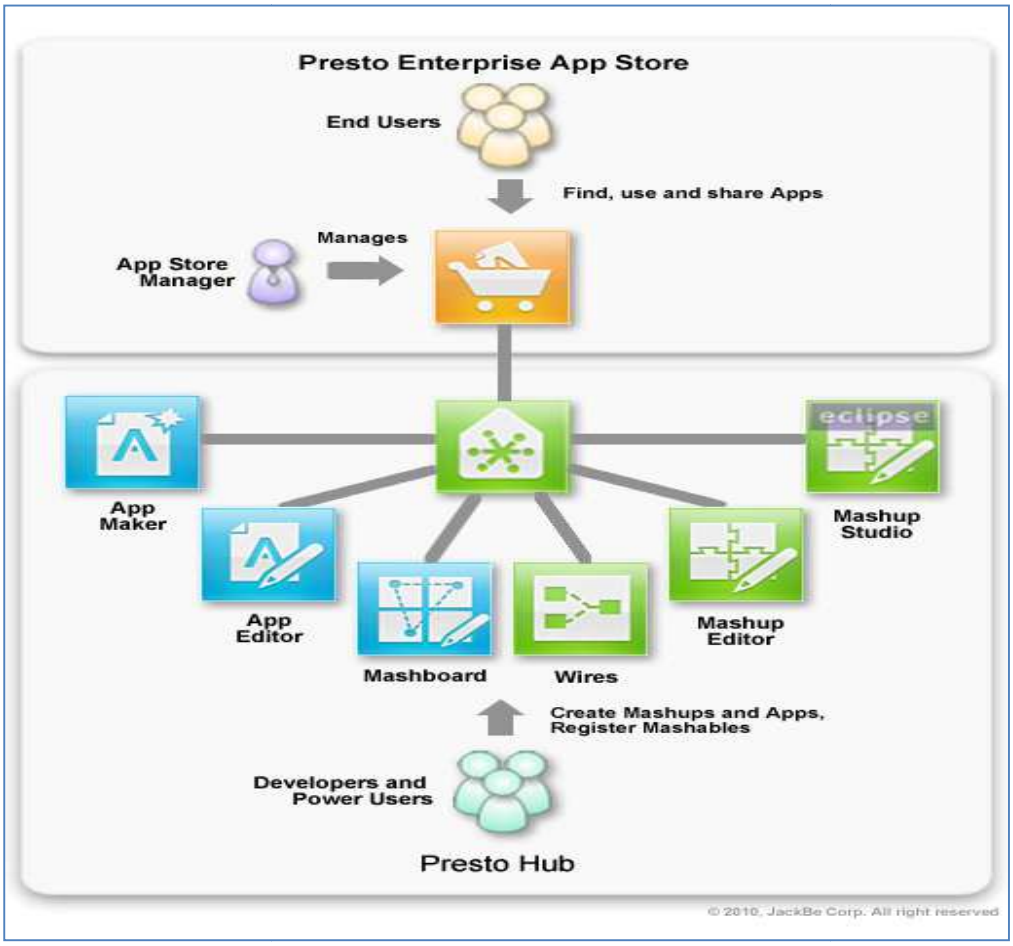

Figure 6: Presto offer

4.2.1. User Interface

Presto allows designing mashup graphically or using EMML composition language. On its graphical UI, mashup can be composed by drag & drop of components. Each component or the whole mashup can be

executed in this design environment; the result is being displayed in structured views (tree or tab) or graphical view (as Maps when it's relevant).

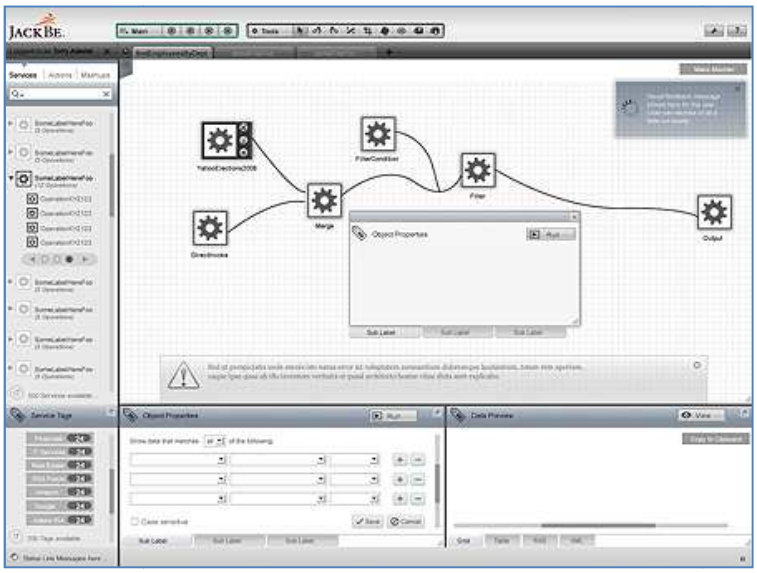

Figure 7: Presto 2.7 Wire interface

On its cloud version (3.0), this interface is similar, but has been enriched by new graphical components, which could be only manipulated through EMML in version 2.7:

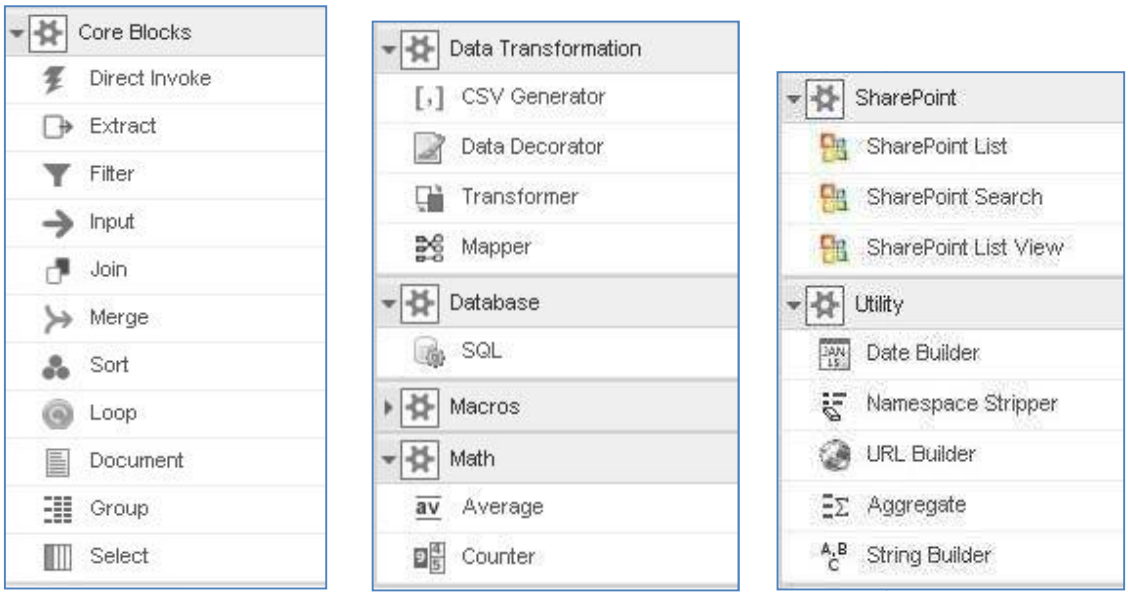

Figure 8: Presto 3.0 base mashups components

Presto allows associating graphical view to mashups and to drag & drop those views in a *Mashboard* interface. Interactions between views can be defined in semi assisted manner.

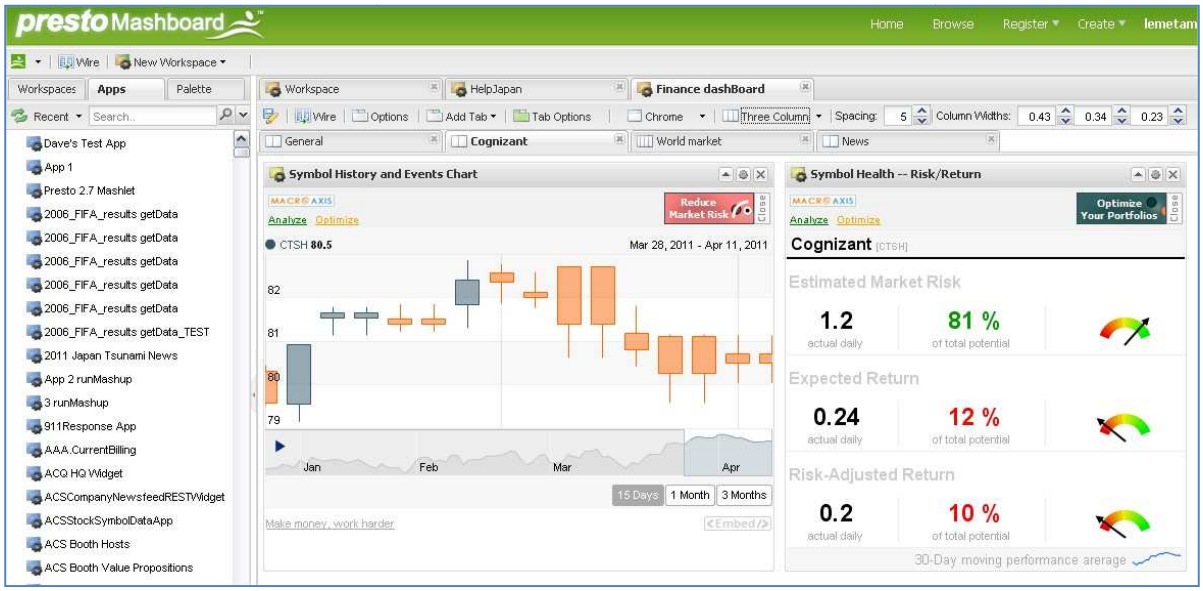

Figure 9: Presto 3.0 Mashboard interface

4.2.2. Composition perimeter

Presto allows data mashing and dedicates many components to achieve this task. Thus, greatly enriched or very selective information can be issued by the mashing process.
Presto offers the ability to mash presentation components, and define wires between designed views.
Presto could allow process design, given the EMML capabilities, though at the moment, there's no interface facility to realize such compositions and the environment is not workflow oriented.

4.2.3. Life Cycle

JackBe is an integrated environment to handle the entire lifecycle on an Enterprise Application. No clear bound can be drawn between phases, as test execution can be made at design time and, for cloud deployment, as a simple switch make the App available for others.

4.2.4. Roadmap

The Roadmap is not available.

4.2.5. Support and Community

The community exists through a slight activity on the forum and blogs.

4.2.6. Technical documentation

User documentation is structured, detailed and quite complete. Contextual help is available. EMML on-line manual covers and explains the whole syntax and use of the language. Installation guide and many working examples are available in the stand alone distribution.

4.2.7. License

Presto cloud version can be used freely by the Mashup Developer Community.

Presto stand alone version is distributed under two types of license:

- **Presto Starter Edition**: can be downloaded by members of the Community and can be deployed in a production environment with the restriction of 5 users and of a production server that does not exceed 2 CPU Cores.
- **Community Edition**: for internal use and Non-Production Use only.

A third party license may be also assumed, for open source project contributing in Presto (as Apache or Spring). For commercial license, there is no information available but commercial contact is displayed.

4.2.8. Functional scope

New base components can't be created with Presto; only predefined ones can be customized and added to the palette of "Mashables". Those components are invocations to Feed, Web Services, files, database, SharePoint lists or Gadgets/embedded scripts. Components assembly is rich on the data field (see Figure 8: Presto 3.0 base mashups components) but is far to implement the whole capacity of EMML.

Applications in Presto cloud are executables as they are designed; they only need to be declared as shared to be used by others. To run Web applications created with Presto out of the cloud, all the Web components must be exported, packaged and deployed in a classic way, that is assumed by the Web App developer. Graphical event principles are implemented in Presto, with events handling allowing multiple Applications of a workspace to catch and react to user events.

Beside, publish/subscribe basic mechanisms are available for Applications to send and consume messages in JMS queues, but no event management is offered for event handling. It may be developed programmatically using Presto APIs. On mashup side, no graphical facilities to handle event is offered. Composing directly in EMML, it may be feasible to handle events, though there is neither dedicated tag for doing this, nor process interruptions mechanisms to delegate event management to external components. Typically, and beside graphical user actions, events in Presto would be used to synchronize asynchronously multiple Applications, as lists or calendars. There is nothing like an event occurring during a process workflow.

4.2.9. Design/runtime separation

In the (commercial) Presto application, whether in its cloud or stand alone version, design and runtime are fully embedded. The OMA consortium offers a separated design/runtime architecture, through a downloadable "reference runtime" which does execute EMML 1.0 scripts.

4.2.10. Data format, export/import facilities, language

Presto applications are standard Web applications; they can be fully exported and reused outside Presto, as they are composed of HTML, JavaScript and CSS. Remote services are bound through REST or SOAP.
XML and XPath are the native language and query language for components in/out and treatment.
Data sources can use JDBC connections or JNDI connection pool.

EMML compositions can be realized out of Presto, in the eclipse-based mashup studio or in any user environment, and then be imported for runtime on the Presto platform, in the cloud or stand alone version. Though, in that case, graphical representation of compositions can't be retrieved automatically in Presto environment.

4.2.11. Standard based architecture, APIs

Mashup Server Connection APIs are available in Java, JavaScript and C# (mashup invocation). Presto architecture is a client/server type, with a Tomcat as application server, and EMML scripts plus connection materials on the client side.

4.2.12. Event APIs

An add-on to Presto Server is the Event Connector, available in Presto 3.1. It makes available a basic connection kit to JMS brokers, allowing implementation of publish/subscribe mechanisms in code developed with Presto APIs.

4.2.13. Possible enrichment and configuration

Presto palette can be enriched with new Applications, Mashups, Mashables (see Functional scope) and Mashlets (graphical components). In the different corresponding palettes, the search is available, but they are basically categorized. No basic components (graphical representation of EMML tags) can be included in the environment.

4.3. WSO2 Mashup Server solution

WSO2 is an enterprise middleware company providing a complete, open source middleware platform. The WSO2 middleware platform is an open source platform-as-a-service for private and public clouds. All WSO2′s products are under Apache License version 2.0 and the company delivers support and services around the products. WSO2 propose an enterprise middleware from the ground up, general purpose Middleware solution, composed with the main following components:

- *WSO2 Enterprise Service Bus* : High performance, cloud ready Enterprise Service BUS,
- *WSO2 Application Server* :
  - Unified platform for hosting Web services and Web applications. The *WSO2 Application Server* seamlessly integrates a number of popular Apache Web services components, for example the popular *Apache Axis2/Java Web services* engine. *Apache Axis2* has extensible messaging engine architecture, so that other *Quality of Service* (QoS) modules can be plugged quickly and easily.
- *WSO2 Data Services Server*:
  - assures ready access to enterprise data scattered across heterogeneous data sources in a unified manner
  - Supports multiple data source types including any relational database accessible via JDBC,CSV, Excel, JNDI bound data sources and Google Spreadsheets
- *WSO2 Governance Registry*:
  - manages the lifecycle of services,
  - finds the dependencies between services and consumers.
- *WSO2 Identity Server*:
  - provides sophisticated identity and security management of enterprise web applications and services.
- *WSO2 Process Server:*
  - The WSO2 Business Process Server is an easy-to-use open source business process server that executes business processes written using the WS-BPEL standard.
- *WSO2 Web Services, Web Services Framework for C, C++ , PHP:*
  - provide for developers the infrastructure to create and consume Web services in their favourite scripting language based on C, the frameworks extend support for other scripting languages such as PHP, C++, Perl, Spring, Ruby, Python and Jython.

- *WSO2 Mashup Server:*
  - Provides a platform for quickly deploying Web service Mashups. Combining simple rich mashups with reusability, security, reliability and governance. The WSO2 Mashup Server offers enterprise-class service composition.
  - Web 2.0-style compliment to the WSO2 Business Process Server, the Mashup Server is oriented towards Web developers (JavaScript + HTML/XML skills) and enables the rapid development and sharing of new services.

WSO2 is a complete platform middleware solution, with components corresponding to each layer of the SOA Stack. In this document, we have focused our study and tests on this product *WSO2 Mashup Server* ***(version 2.2.0)*** . WSO2's Mashup Server is aimed at Web developers seeking a complete environment for building, deploying, and administering composite applications. WSO2 Mashup Server help developers to create Mashups, and for those with an understanding of JavaScript, XML, and AJAX, this toolset makes developing Mashups simple. However, the tool is not targeted towards Business users.

WSO2 Mashup Server consists of a framework for use on both the client and server side of a Mashup. On the server side, deploying a Web service can be as simple as writing the service in JavaScript and moving the file to the proper directory. All the necessary resources to use the service from a client are automatically generated, such as the WSDL and schema, along with a number of other useful artefacts. The server is based upon WSO2's WSAS (Web Services Application Server), widely tested for interoperability, so the services should be accessible from most any client.

4.3.1. User Interface

This paragraph describes how to develop a basic Mashup with the help of the WSO2 Mashup Server. The Mashup design is mainly JavaScript oriented. The user connects to the WSO2 Mashup Server and then adds a new Service, with a 3 steps process: creation, edition and deployment of the Mashup.

1) Creation of the New Mashup *My Mashup (Manage>Services>Add)*

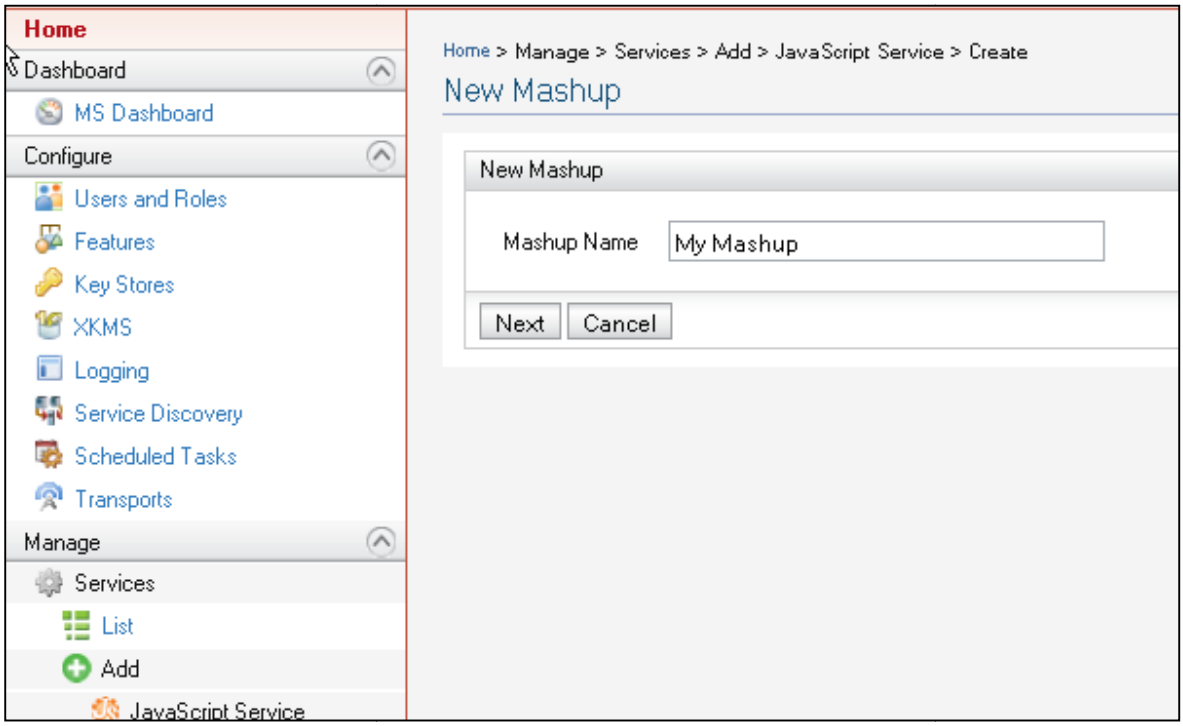

Figure 10 : Mashup Creation

2) Editing the Mashup (JavaScript) *Create> Edit Mashup*

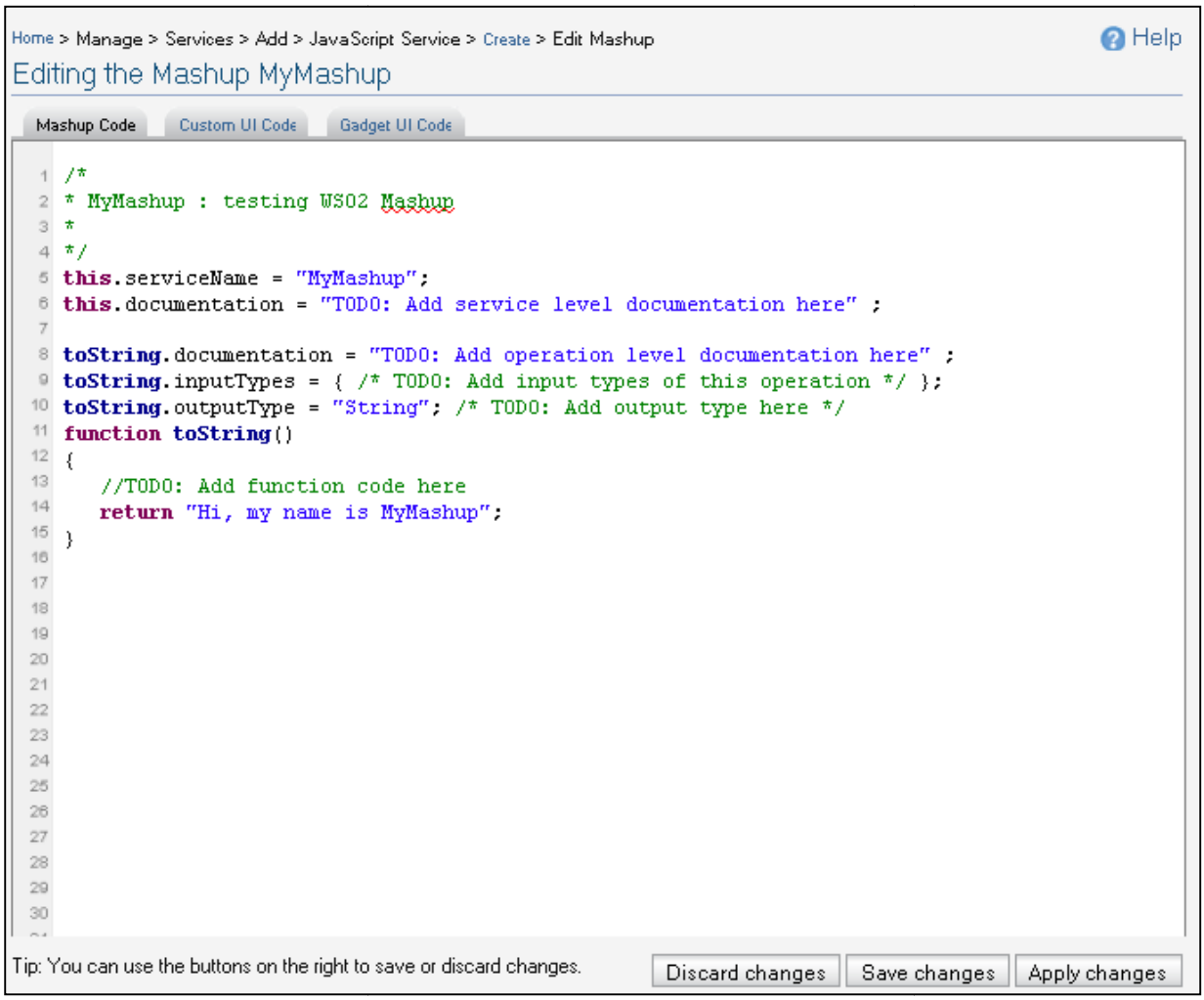

Figure 11 : Mashup Edition

3) Saving and Deploying the Mashup (deployed automatically after saved)

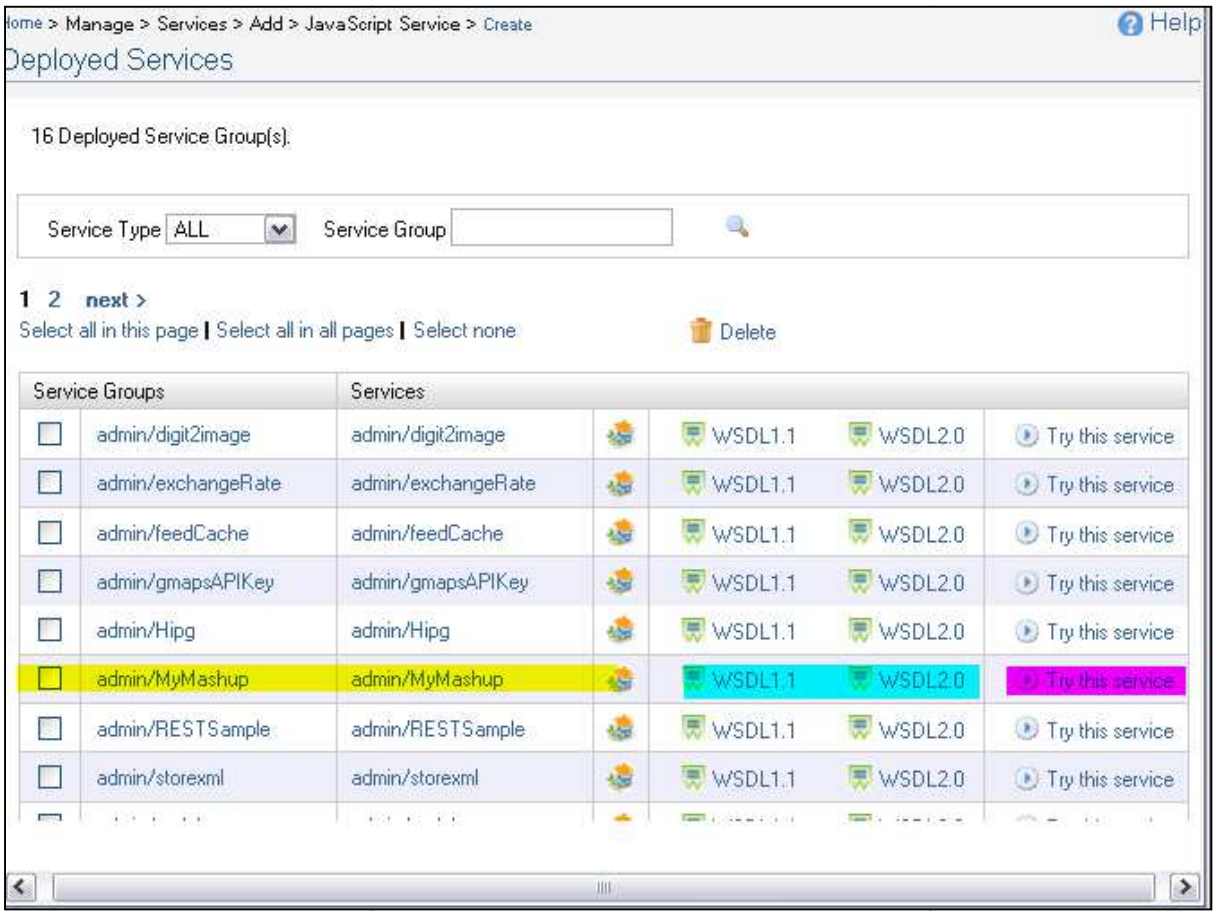

Figure 12 : Mashup deployment

In the example, The deployed Mashup is deployed for the user *admin* and available under the name admin/MyMashup. It is worth to notice that the user may directly call the Service to try it by using the basic User Interface (See next figure).

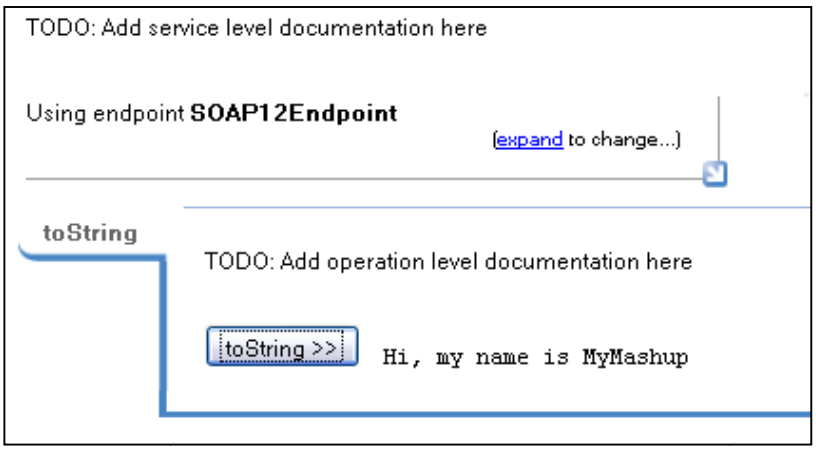

Figure 13 : Mashup UI

WSDL description is also available and it is possible to connect WS consumers applications compliant to WSDL 1.1 or WSDL 2.0 standards (see paragraph 3 before) .

4.3.2. Composition perimeter

The core is pretty complete but in terms of UI it is developer oriented. A Drag & Drop Mashup tool is really missing, even if there is a commercialized eclipse plug-in.

4.3.3.Life Cycle

The WSO2 Mashup Server allows you to acquire web based information from a variety of sources such as Web Services, Web pages, Databases or Feeds, combine it in interesting and useful ways before exposing the result again as a Web Service or page, Feed or a Google gadget. The Mashup can even interact with users via e-mails, instant messages and desktop alerts. The whole life cycle is covered: creation of forms, creation of component, assembly and finally deployment.

4.3.4. Roadmap

There is no clear roadmap for the WS02 Mashup Server. There is a clear roadmap for the WSO2 ESB product with a reference to a link between WSO2 ESB and the Esper CEP. (http://www.slideshare.net/prabathsiriwardena/complex-event-processing-with-esper-and-wso2-esb)

4.3.5. Support and Community

As this is a project under apache organization it leverages the support aspect with a large community. A forum is dedicated to the WSO2 Mashup Server with a moderated activity.

4.3.6. Technical documentation

The documentation is clearly split in the traditional way: Installation, User, Programmer, and Administrator. There is no surprise here. All the documentation is well done and additional support is given in the forums.

4.3.7. License

WSO2 Carbon and the products built upon it are 100% open source and released under Apache License 2.0.

4.3.8. Functional scope

The WSO2 Mashup Server acts as a hub for Web 2.0-style interaction and development. Providing a simple way to deploy services developed in JavaScript, the Mashup Server offers access to REST and WS-* web services, feeds, and scraped web pages. This data can be scripted together quickly using common Web developer skills and the result exposed as a new service – or a web page, gadget, email or instant message communication. The Mashup Server is an ideal platform for defining composite services for use in user interfaces and mobile applications.

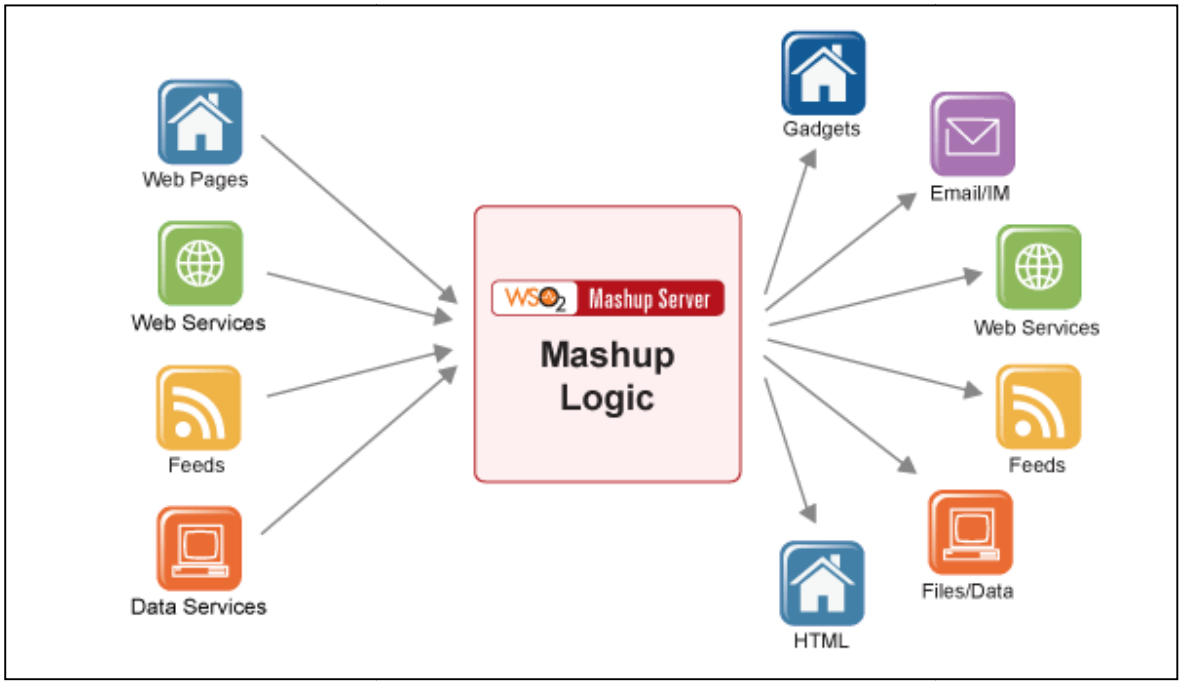

Figure 14 : WSO2 Mashup logic

4.3.9. Design/runtime separation

There is no clear separation between design/assembly and runtime. All the life cycle is done through web forms.

4.3.10. Data format, export/import facilities, language

Technologies: XML, WSDL, REST, HTML, JavaScript, Java, BPEL. The mashup part is mainly JavaScript oriented.

4.3.11. Standard based architecture, APIs

The Google Gadget API (HTML/JavaScript) is supported by the WSO2 Gadget Server.
Java APIs for the Governance Registry, to add, get and manage registered resources (services and files)
Web Service API for C and C++ (WSF/C), allow C applications Web Service integration.

4.3.12. Event APIs

The Esper event environment as already been linked with WSO2 ESB in a kind of mockup but there is no really development or product based on this link between both products

4.4. IBM WebSphere sMash

WebSphere sMash introduces a simple environment for creating, assembling and running applications based on popular Web technologies. The tool allows the service composition or the aggregation of RSS feeds. A cloud version is also available, which is offering the same functionalities than the stand alone version.

4.4.1. User Interface

sMash environment includes the Application Builder module, a browser-based IDE which enables visual assembly-style development by drag & drop operations.

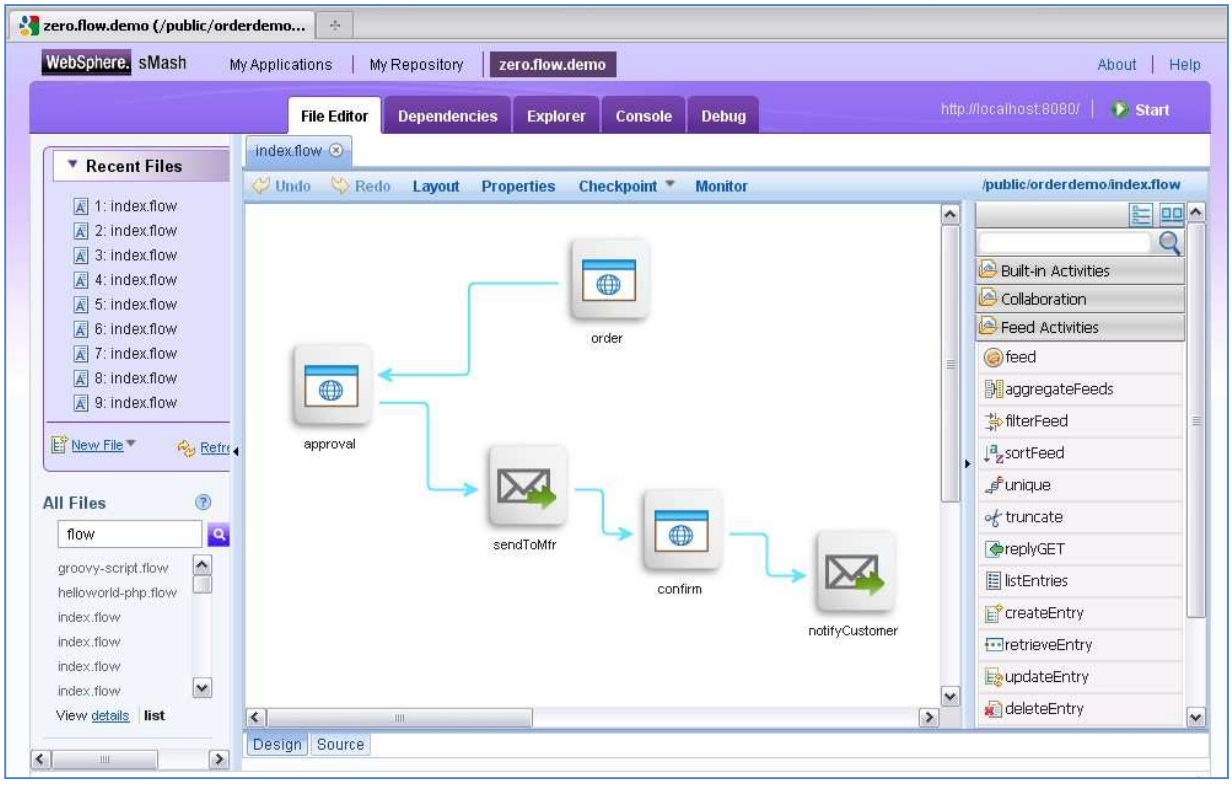

Figure 15 : Smash graphical composition environment

In this environment, business user can compose workflows, with few development notions. Though, setting components or link parameters needs some understanding of development principles. For advanced Web development, App Builder also provides basics IDE's functionalities. Otherwise, an eclipse plug-in can be used.

4.4.2. Composition perimeter

Business approach: workflow can be composed, including human interaction, orchestrations of REST Web Services. Though, native components are Web & data oriented, so one's own business services have to be developed. Components allow loops and conditions can be added to links, but beside this, workflow sophistication is limited. Therefore, sMash can be considered as process oriented platform including data aggregation.

4.4.3. Life Cycle

The full life cycle is covered:

- component creation, PHP and Groovy based
- service composition/orchestration through a easy-to-use user interface
- deployment

Test modules are provided for native application components, but no particular facilities for custom components.

4.4.4. Roadmap

sMash roadmap is not currently[8] available because their release planning process has been moved and is currently located behind the IBM firewall. IBM says they're working on externalizing RTC, but there's no date associated to this task.

4.4.5. Support and Community

WebSphere sMash has an associated incubation project, Project Zero, with an associated community offering discussion forums, blogs, lot of code samples, tips, demo videos, etc. The corresponding sites are actives and information proposed is useful. The forum is animated by project zero developers, which are very prompt to answer any question.

4.4.6. Technical documentation

Documentation is moderately structured, organized by "action themes" (creating, debugging applications), associating an article to each of theses themes rather than describing each function or possible operation in the environment. Therefore, the documentation only partially covers the needs of information. The documentation is on-line only.

4.4.7. License

A Commercial license must be contracted to use sMash in a production environment. Otherwise, sMash offers a development License based on the public incubator "Project Zero". Under this license, the source code is downloadable and the program can be used in a non-production environment, for internal evaluation and for the development, demonstration and testing of application programs.

4.4.8. Functional scope

New components can be created, whether in development mode, or by aggregation of others components. Graphical component assembly is the function dedicated to the App Builder module.

[8] March 2011

The environment development and aggregation is built on top of WebSphere application server, and applications are easily deployed within. No event principle is implemented unless considering as an event an http request trigger the start of the application. Managed event are GUI events.

4.4.9. Design/runtime separation

Runtime is embedded with the design as a whole platform. The possibility has been evoked in the community to produce war applications to be deployed in any application server, though the idea didn't lead yet to a real plan.

4.4.10. Data format, export/import facilities, languages

WebSphere sMash is based on the following Web technologies:

- A dynamic scripting runtime for Groovy and PHP
- Application programming interfaces optimized for producing REST services
- Rich Ajax Web user interfaces
- Integration mash-ups and feeds

Supported Technologies: java/groovy, WSDL, REST, HTML, JavaScript, JSON, RSS, XML, PHP.
Applications created in sMash can be packaged and exported and stored on a Web site. When the site is declared as a sMash repository, packages are accessible from any other sMash for import.

4.4.11. Standard based architecture, APIs

Applications created in sMash can be executed remotely, as they're exposed as REST services. Also they can dynamically provide any REST response content. Within the environment:

- Database native APIs are available[9] for: IBM® DB2®, Apache Derby, MySQL, Oracle and Microsoft® SQL Server®.
- Remote connection to any HTTP or HTTPS APIs can be established through the java *zero.core.connection* package. FTP connection can be established as with the package.
- JavaScript, Java and Groovy feed API are offered to simplify feed contents access and manipulation.

4.4.12. Event APIs

The concept of event as it is evoked in sMash has to do with application interactions or graphical user event, not with external event that could occur during the application execution. Though HTTP event can be implemented, handling of those events can't interfere in the back-end process execution. Native components don't allow interruption and loop components aren't conceived to wait permanently for events to occur. There is no available API to interconnect an event server. Though, we must note the availability of a Java messaging API, which offers facilities to connect to a JMS broker.

[9] All supported database with version :
http://www.projectzero.org/sMash/1.1.x/docs/zero.devguide.doc/zero.data/overview.html

4.4.13. Possible enrichment and configuration

The component palette can be enriched with new components, basic or composite (issued from a composition). Though there's no automatic process to achieve it, a developer should write configuration files and provide them at the proper locations.

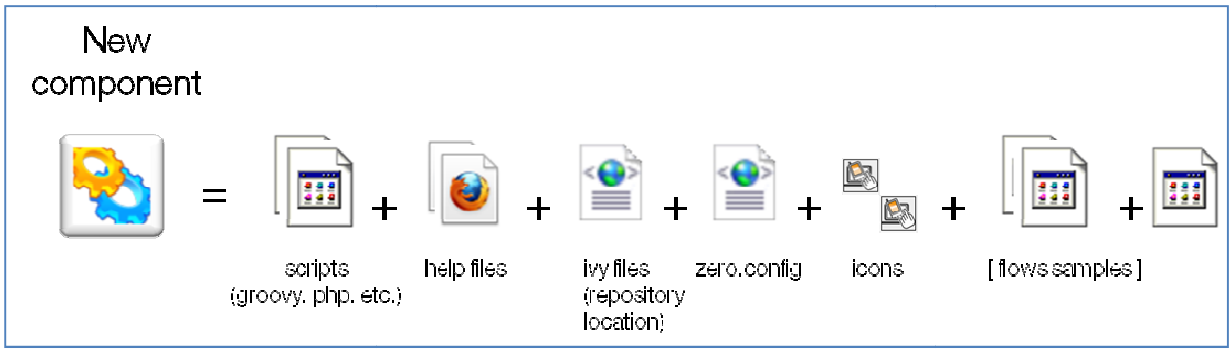

Figure 16: component creation with sMash

In the palette, components are categorized, but can't be organized in a hierarchy nor tagged.

4.5. Cordys MashApps Composer

Cordys Process Factory is a Platform as a Service (PaaS), accessible through a Web browser, for business application development in the Cloud. The platform contains a MashApps Composer, an on-line modeling environment that allows the creation of new applications through forms-driven business processes design. Cordys provides a market place to buy or offer MashApps applications.

4.5.1. User interface

The MashApps Composer is a browser-based environment, where all creation operations are done graphically, by drag & drop, tree selections. Business Processes are created by drag & drop from a palette of elements and activities:

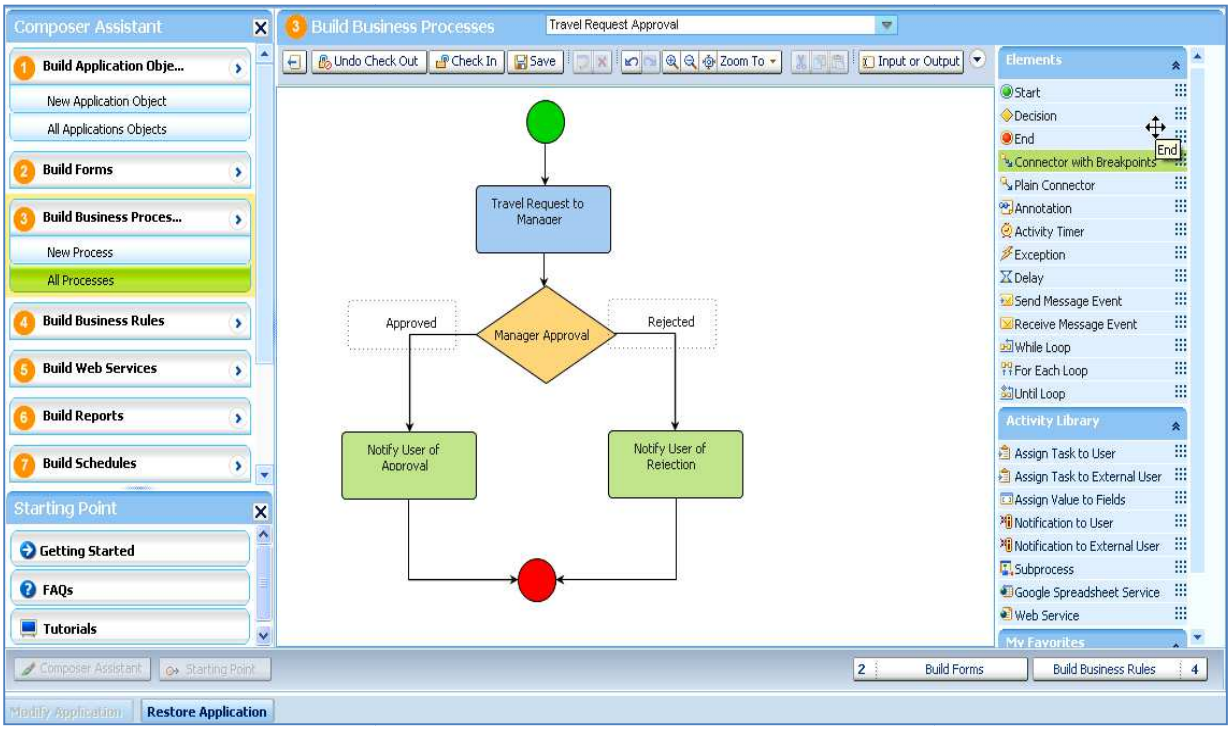

Figure 17: business process design with MashApps Composer

As well, forms are designed by drag & drop of elements from a palette to the cells of a grid:

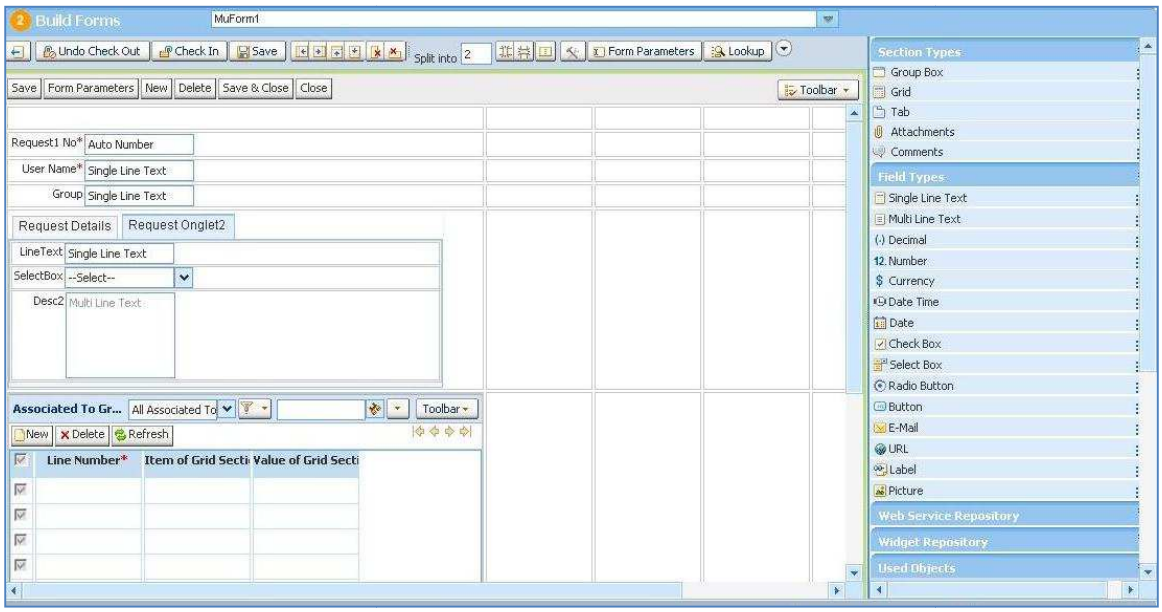

Figure 18: form creation with MashApps Composer

4.5.2. Composition perimeter

The tool achieves presentation composition, assembling classical forms, widgets, Google gadgets, in a graphical and user-friendly manner. It allows data interactions between presentation components, but is not much conceived to mash data from different sources.
The composition of process is offered, through a basic BPMN-like notation, allowing rather simple operations. Available Elements and Activities are shown below:

Figure 19: Elements & Activities of the Business Process modeler

4.5.3. Life Cycle

The full cycle is covered: component creation, application design, deployment and runtime in the Cordys cloud. No validation module is offered.

4.5.4. Roadmap

No information is available.

4.5.5. Support and Community

There is a community space where support is provided by Cordys team. This forum is moderately active (about 2 new posts per week, rather quickly answered). The community section allows user to post to Cordys team, asking for support, reporting issues or giving feedbacks, but no direct exchange between members is supported.

4.5.6. Technical documentation

The on-line documentation is structured and covers all users operations in MashApps composer. There's no contextual help and the "search" mode is not quite effective. Downloadable "Quick Start" tutorials guide the user through all features of the composer, in an easy and detailed step-by-step way.
The available FAQ contains very few questions.

4.5.7. License

Cordys use is available under commercial license.

4.5.8. Functional scope

The *MashApps Composer* allows modeling Business Processes, based on BPMN-like notation, creating forms, inserting widgets, target Web Services or databases, and associating processes with the graphical or data objects defined for execution. Components that can be created are forms and application objects (data storing), though, the environment is not oriented for the re-use of those components. New widgets and Google gadgets can be added in the palette, based on the public widget URL or script, but new defined widgets are accessible only in the application they've been defined. No principle of complex or external event is implemented in the environment, only user or internal application events.

4.5.9. Design/runtime separation

The environment embeds design and runtime in the cloud.

4.5.10. Data format, export/import facilities, language

Data and languages used in Cordys are proprietary, though, even XML seems to be used for some components descriptions, there is no access to it. Developed applications can be packaged and exported to a zip file, which can be imported by other users of the composer in the cloud. Also, the application can be proposed for selling on Cordys market place.

4.5.11. Standard based architecture, APIs

SOAP Web Services can be generated by the composer, and SOAP & REST Web Services can be registered for use in forms.
Generated SOAP Web Services may be accessed by other users of MashApps Composer in the cloud, but it seems impossible to reach them from a basic client from outside the cloud. Beside that, no other input or output API is available[10].

4.5.12. Event APIs

No event API is available, nor a mean to interconnect with an event engine.

4.5.13. Possible enrichment and configuration

As explained in the “Functional scope” section, component or composite creation to enrich the tool’s palette is not the principle of Cordys platform. Rather it is to create new applications, useful at a particular business, for sharing with others, directly or through the market place in the cloud. So of course, little can be added in each application palette, let alone in a hierarchical or a ranked way.

[10] MashApps for smart phone hasn’t been studied, possibly this part would be more open.

4.6. Bonita Open solution

Bonita Open solution is a very smart and flexible BPMN design tool. It combines a studio for process modeling, with a BPM & Workflow engine, and a very handy user interface.
We did our test on the Bonita Open Solution 5.4 which was released on the January 2011. This new version offers key feature upgrades to achieve greater usability. Bonita Open Solution was downloaded more than half million times increasing the community. As far as we have seen the Bonita soft solution seems open to business user as well as developers. Without going through any development, the user can obtain his application. Of course, if you need more complexity you can go in the integrated development environment. The easiest part is the One click multi-environment deployment making this aspect a lot easier.

4.6.1. User Interface

The UI is quite business user oriented, so very easy to use with a drag&drop style. The solution offers a WYSIWYG forms editor and many connectors to easily interact with other systems. The workspace does look like a web2.0 webmail where the user can find his main board, tasks, process, messages, so a very smart interface with a zoom panel that make it easier to manipulate big designed process.

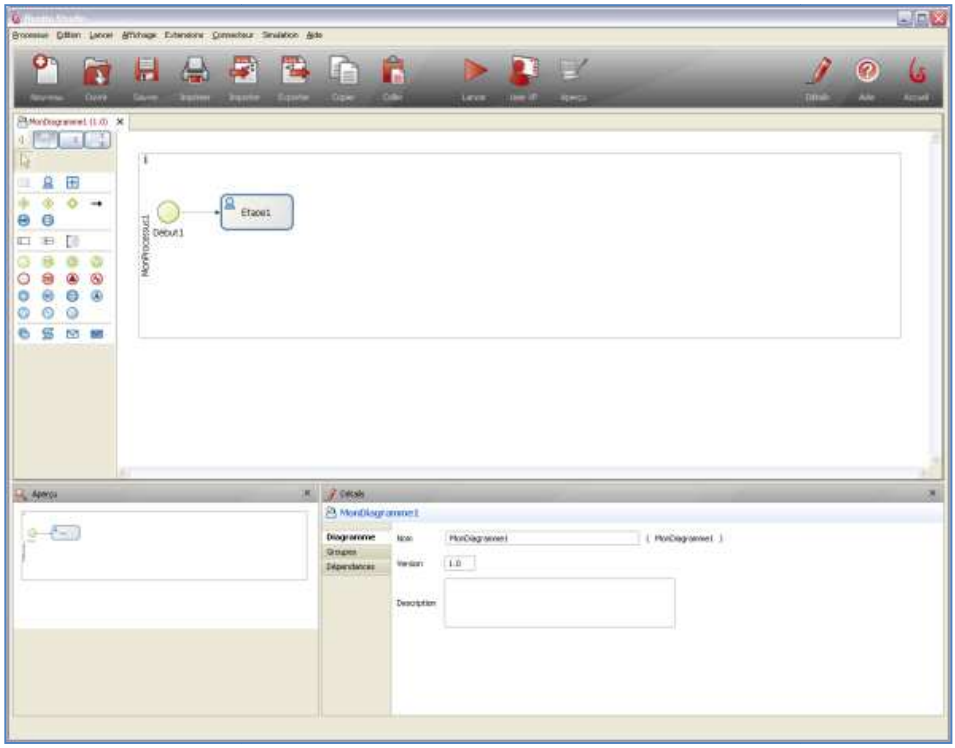

Figure 20: BPM design view

4.6.2. Composition perimeter

With this drag & drop style the tool provides a technology to handle business processes through a user friendly and easy-to-use interface. Drag & drop widgets creation including text inputs, drop-down lists, radio-buttons and so on, to create user forms corresponding to the human steps of the process.

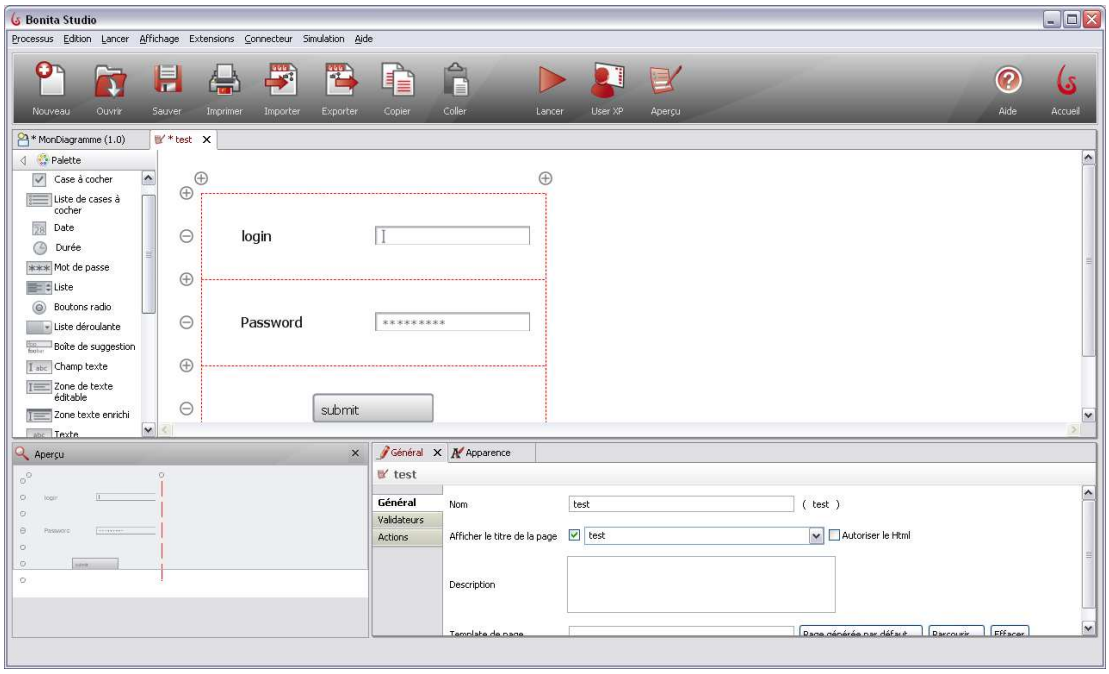

Figure 21: forms creation

The Bonita solution has a BPM People Centric approach where tasks are routed from one person to another. The process is fast to make and easy to deploy. This aspect has been very well thought. The solution's editor is quite liquid and handles big process without getting lagged.
The Interface creation through the fulfilled palette becomes very handy; in addition the community shares templates for UI reusability. So, we get very nice results in short time.

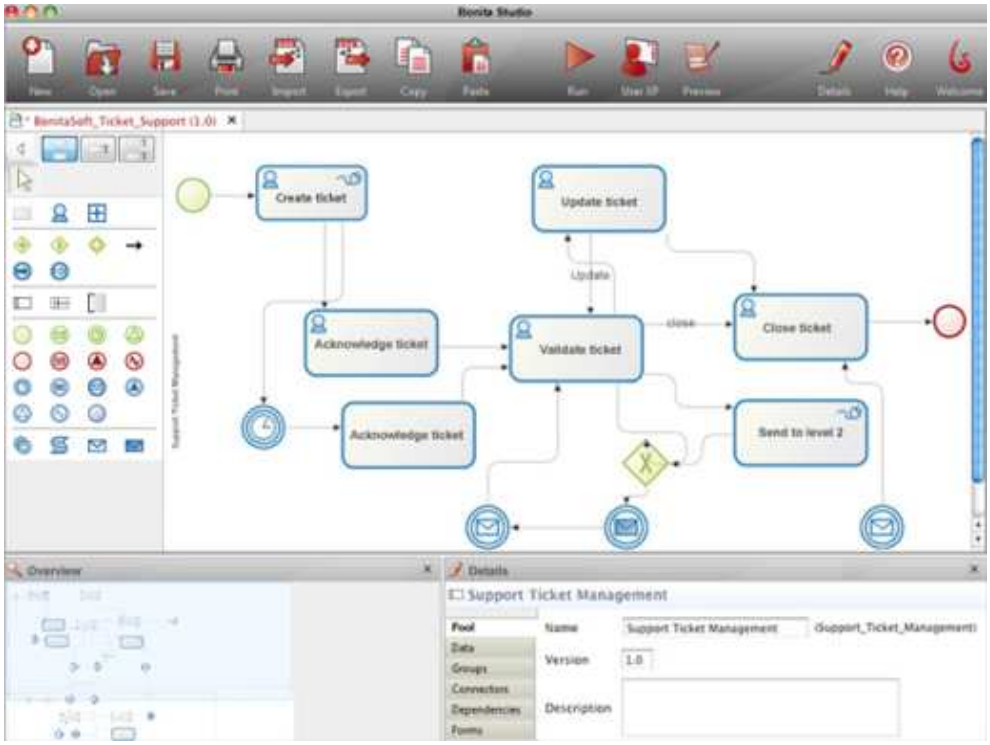

Figure 22: heavy process display

4.6.3.Life Cycle

Creation of component, assembly and deployment are managed through the platform. This aspect has been clearly worked to be as easy as possible, to allow functional people to create and deploy their workflow without any specific knowledge. In terms of People centric workflow, Bonita offers a full optimized solution.

4.6.4.Roadmap

The roadmap is very detailed as we do have access to the development roadmap instead of a simple communication roadmap. The 5.5 version is in progress and they seem on time regarding their sizing.

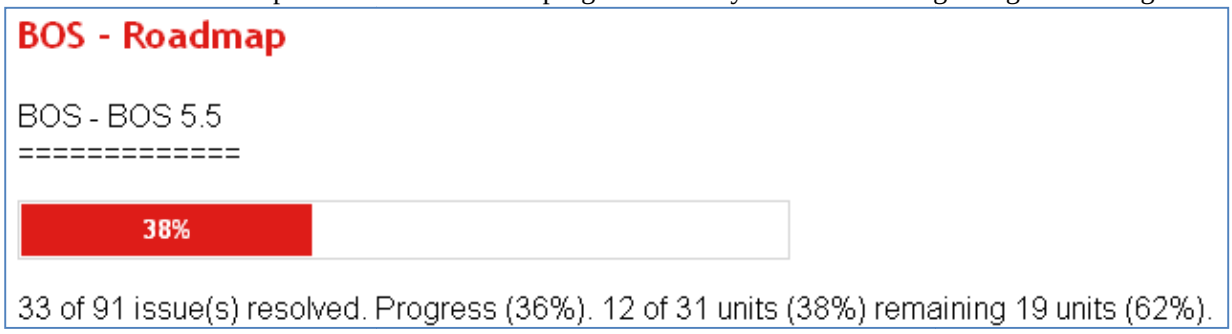

Figure 23: Boita Soft 5.5 version roadmap

4.6.5. Support and Community

Blog and forum are currently active (3000 forum registrations). The full documentation and tutorials are available online as well as a contribution section where everybody can post project or homemade connectors. The source is accessible online and it is possible to report bugs within the bug tracking section of the website.
The whole support has been integrated and can be accessed through the standalone application.

Figure 24: Application Home page

4.6.6. Technical documentation

Thanks to the community the website is full of documents such like video tutorials but also technical documentation about installation, protocols, connectors and API documentation.

4.6.7. License

Licensed under GNU General Public License v2:
the GNU General Public License is intended to guarantee your freedom to share and change free software and make sure the software is free for all its users.

4.6.8. Functional scope

We could imagine adding our contribution with an EPL or CEP connector. The website is full of documentation to explain how to build a new connector. Taking the inspiration from others developed contribution it seems feasible to do so. Bonitasoft is also offering to export the application in a jBPM3 format. jBPM is no longer an isolated process engine but your processes can be combined with powerful business rules and complex event processing in the jBPM5 version. In parallel there is a migration project to enable the migration from jBPM3 to jBPM5.

4.6.9. Design/runtime separation

At design time the solution offers to design processes through the WYSIWYG interface.
Then we can define the data through forms and link our application process to those data.

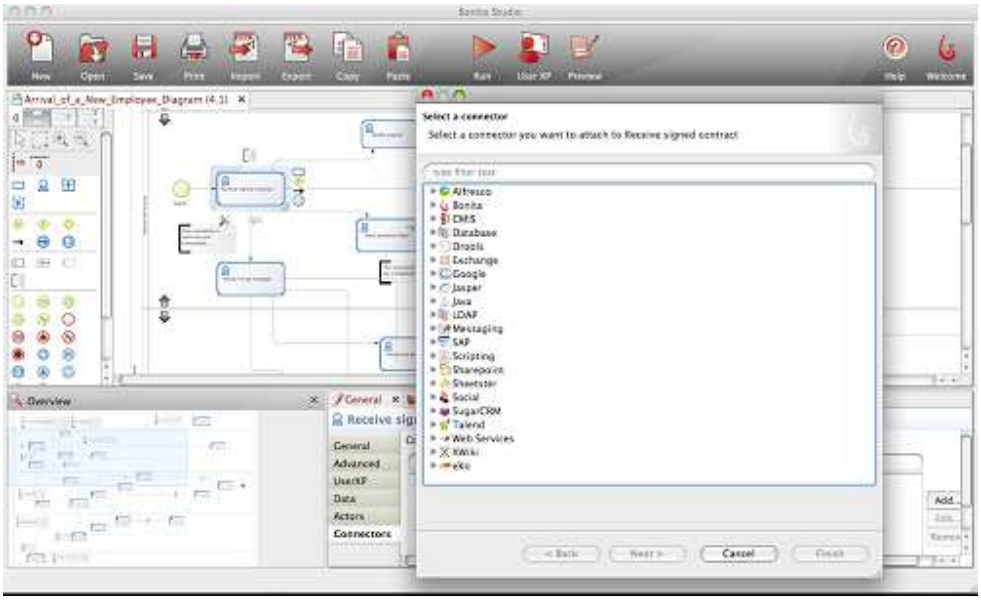

Figure 25: external data access definition

We can also link our application process to other system or remote application through provided connectors. After those binding we have to create the user interface through a WYSIWYG form editor which offers a preview mode.

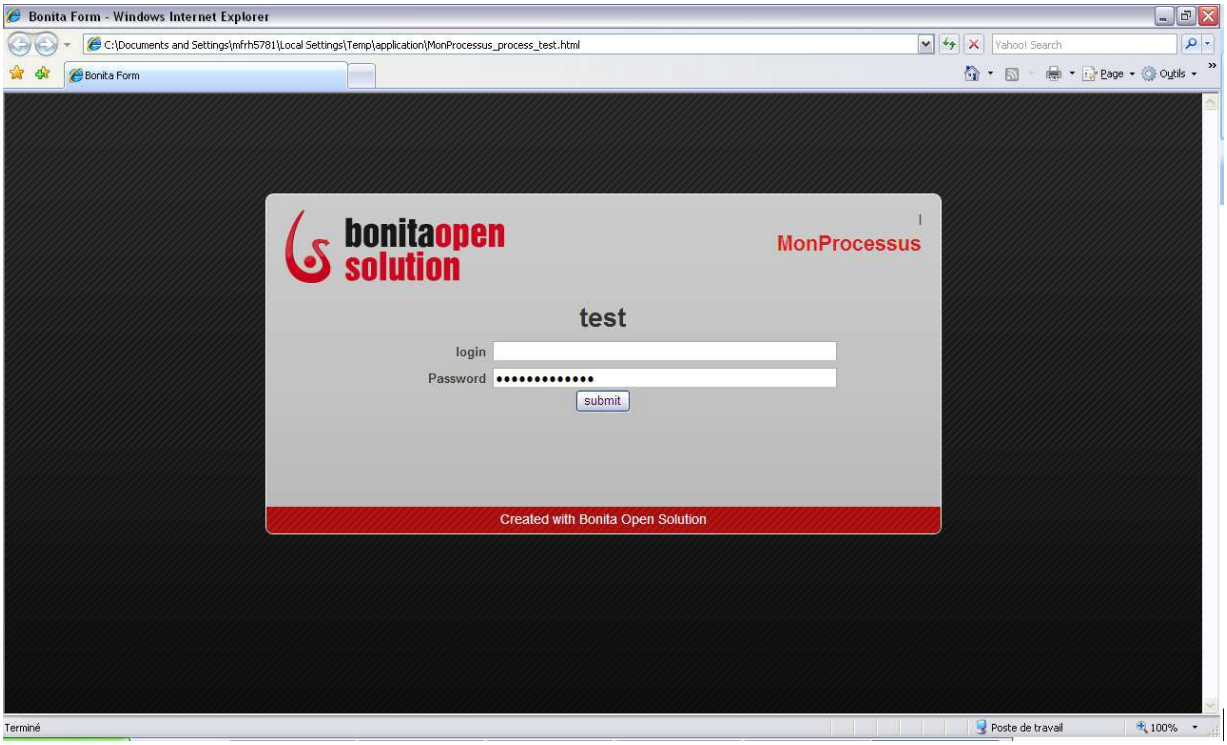

Figure 26: preview mode

We can finally deploy the application in one click or generate a war file for a future manual deployment. At runtime the Bonita solution offers the ability to analyze report, check the process completion, plus an administration section to manage tasks and cases.

4.6.10. Data format, export/import facilities, languages

Bonita Studio comes with 100+ built in and contributed connectors for many commonly used commercial and open-source databases, messaging, ERP, CRM, ECM, but also with other applications such like Facebook, Google calendar, Amazon, Paypal or Twitter.

Figure 27: connector selection view

In terms of language it is mainly groovy/java oriented, the applications can be exported as a webapp and automatically deployed in a jetty server. The application process can also be exported in a jBPM3 and BPMN2.0 format.

4.6.11. Standard based architecture, APIs

Java API allowing to pilot processes definition, deployment, execution and monitoring, instead of using the tools provided with the solution (BOS Studio and BOS User Experience).
To access the Bonita runtime remotely, the API is also available in REST or EJB mode.

4.6.12. Event APIs

There are some event connectors such like signal event connectors. The platform allow to design processes that can be triggered by an external event (e.g. from another process). So using catch signal or catch message the application can trigger some processes related to those events.
The website does contain some sample such like a “Temperature sensor simulator” where a process checks the temperature sensor and throw a message to another process when the limit is reached.

4.6.13. Possible enrichment and configuration

As this is an open source project it is always possible to extend the whole solution. The website is full of documentation of how to install and configure the solution. Otherwise, the basic installation and configuration is very easy and it is though to be customized in terms of architecture such like the type of server or security protocols.

4.7. GEasyBPMN Editor

Geasy BMNEditor is an open source, web-based BPMN 2.0 editor developed with GWT.

4.7.1. User Interface

Like Bonita, GEasyBPMN Editor is very easy to use with a drag & drop style.

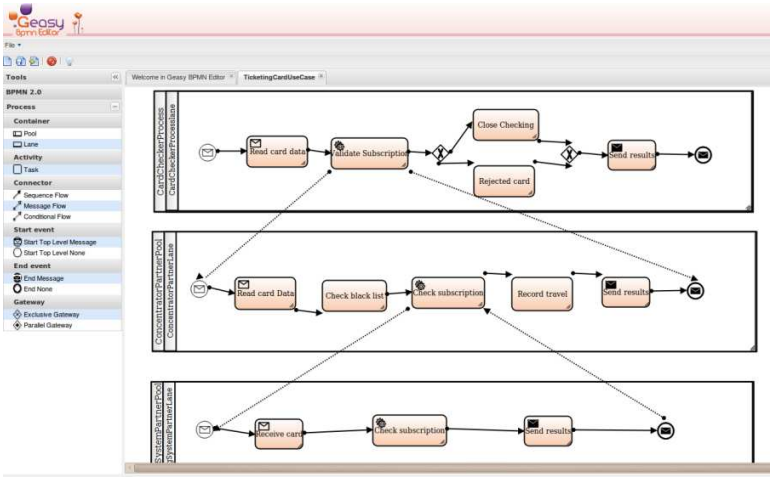

Figure 28: GeasyBPMN user interface

4.7.2. Composition perimeter

The tool provides lot of BPMN 2.0 concepts to design collaboration, human workflow or technical processes. To realize these kinds of processes through a drag & drop style.

4.7.3. Life Cycle

Creation of component, assembly and deployment are managed through a set of tool. GEasyBPMN Editor is used to design Workflow. PEtALS Studio is a tool based on eclipse able to achieve the assembly. PEtALS ESB is in charge to deploy the process.

4.7.4. Roadmap

The next roadmap planned before the end of the year will contain:

- Add choreography tasks.
- Add all events concepts: Start event, End event, Intermediate event, Event handling… etc.
- Add data associations

4.7.5. Support and Community

Mailing list is currently active. The full documentation and tutorials are available online.

4.7.6. Technical documentation

The technical documentation is available at this address:
http://research.petalslink.org/display/geasybpmneditor/GEasyBPMNEditor+Overview

4.7.7. License

GEasyBPMN Editor is distributed under GNU AFFERO GENERAL PUBLIC LICENSE

4.7.8. Functional scope

The main features of Geasy BPMNEditor are:

- Allows users to create and edit BPMN 2.0 diagrams.
- Import and export of the created diagrams to a BPMN 2.0 XML file.
- Import and export of the created diagrams to a XPDL 2.1 XML file.
- Export of the created diagrams to a BPEL 2.0 file.

4.7.9. Design/runtime separation

Design and runtime are clearly separated. GEasyBPMN Editor is used for the design time and PEtALS ESB for the runtime.

4.7.10. Data format, export/import facilities, languages

GEasyBPMN Editor allows to import or export XPDL 2.1 or BPMN 2.0 files.

4.7.11. Standard based architecture, APIs

The tool is based on standard. The collaborations are designed and exported into BPMN 2.0 files. These files can be converted into BPEL that can be deployed on PEtALS ESB.

4.7.12. Event APIs

The handling of events is planned in the next roadmap.

4.7.13. Possible enrichment and configuration

5. SocEDA Mashup requirements

The tool should allow graphical sequencing of back-end and front-end components, in order to build complex components (composites) or composite services (workflows) including events.
An option would be for the tool to produce simple widgets, configurable by basic users, to customize previously created application at their particulars needs.

5.1. User Interface

The mashup designer should be a user-friendly tool, using drag & drop principles to create composites or workflows, and select box or text box to configure components.
See § 5.8 - Functional scope for details of the interface needs, depending of the operation type implied.

5.2. Composition perimeter

The design tool should be able to design workflow. The tool should be able to generate presentation components implied in the workflow (as human validation requests).
It could be useful that the tool allows to create presentation components, and to mash them in a complete user-interface, apart of the workflow, but to offer additional functionalities, or customizations. Though, in that case, deployment binding with those users interface should be available or cost a few to develop.

The need to aggregate data is patent in order to mix information from several events, though it's not what enters usually in the field of data-mashing (it rather implies feeds mashing or so). So, including data in the requested composition perimeter may be tricky, and could be pointless. Ad hoc mechanism or the possibility to develop specific extension may be the need to provide those particular data mashing.

5.3. Life cycle

The design tool may cover only the design phase, modeling components, composites and workflows, if a standard runtime package can be produced (ex: producing a .war file that can be deployed in a application server), or any proper deployable file for the SocEDA platform's identified workflow engine.

If not, and if no separated design/runtime tool can be selected, all phases must be covered, design plus deployment and runtime.

5.4. Roadmap

Depends on the tool functionalities and maturity, we may base on its potential, through the roadmap planned.

5.5. Support community

As we know for sure no existing tool will fulfill all our requirements, we expect it is supported by a large and effective community. It would be a certain guarantee for product progressiveness, for the requests to be addressed and for the component library enrichment.

5.6. Technical documentation

Obviously, a technical documentation structured, complete and offering plenty of samples would be preferred. Though, a lack on this domain can be compensated, if the tool is intuitive (for the part we will use it) and if the community is responsive and productive (for the part we may develop on it).

5.7. License

Permissive open source licenses are preferred, though, other types may be considered until a free access to the software is possible.

5.8. Functional scope

The mashup designer should support to be enriched by new atomic abstract components, even if the work to achieve this implies developer skills. The designer should be able to configure abstract atomic components, to make them concrete (runnable), through graphical manipulations.
It should be able to design composites (components aggregations), and workflows (components orchestrations), through graphical manipulations. This should be possible using concrete components, or abstract ones that could be configured at the design time. The tool should be able whether to deploy a workflow as an application or to produce a deployable package/file.
The tool should support “start events”, “end events” and “intermediate events”, and handle the concrete link with an event engine, whether the event interactions at runtime are managed inside or outside the tool.

5.9. Design/runtime separation

Design and runtime are preferred to be separated, though, if import/export functionalities needs are covered, both can be embedded.

5.10. Data format, export/import facilities, language

The design tool should be able to produce a file in a standard format (as XML) describing components, whether they’re atomic or composite, workflows (plus applications in the case they can’t be defined as a workflow, because including complementary configuration or properties).
The tool should allow export/import of those description files, and so in particular the application exchange between users.

5.11. Standard based architecture, APIs

The design tool should be able to invoke HTTP, in order to instantiate at the deployment phase the previously designed CEP components of a new application. If not, the tool should be extensible so to add this facility. The design tool should be able to invoke an external API, preferably a standard parser (as JaxB), for customized coherence control. Graphical API (JavaScript, Dojo, or others) would be useful to be able to integrate customized design components.

Whether the design tool’s associated runtime is embedded or separated, it should be able to bind SOAP Web Services. About the design part only, if widgets are generated, REST call should obviously be possible. There’s no particular need for an API to invoke the design tool.

5.12. Event APIs

Runtime should whether allow natively handling incoming events, or offer a standard access to be requested. Otherwise, it should provide an efficient interface to queuing servers, as far as possible standard servers (as JMS). For outgoing events, an API toward a JMS server would be preferred.

5.13. Possible enrichment and configuration

The design tool should allow enriching its palette of components. This operation should be dynamic and user-friendly for composites (components composition) or for newly made concrete components (abstract components once parameterized). For new abstract components, basic import or other manual operations would be enough.
The palette should be categorized and would be preferred hierarchical. Thematic ranking or other type of ranking, search engine, would be useful options.

6. Conclusion

This document describes a state-of-the-art mashup design toolset, including a BPMN solution. During this study, we reviewed very different kind of mashup solutions, and pushed our tools examination to process orchestrator, as we could confirm mashup tools tendency to overlap the field of BPM. Indeed, mashup composers do allow process design, in addition to data and presentation components assembly. They allow more than end-user Web sites or lightweight portals designing. Though, when they offer the simplicity of design, they appear too limited to include external events in their compositions. On the contrary, BPMN/BPEL tools natively allow introduction of events in workflows design, but they are based on a quite sophisticated notation for an end-user.

At this stage, we did not find an existing designer allying the simplicity of typical mashup tools to the sophistication required to introduce events in compositions. Not to mention Complex Event Processing fully integrated in process workflows.

Complex Event processing is attracting more and more attention in the IT community. The current CEP solutions have been based on technical and formal languages targeting the developer's communities. We would like to turn this aspect as easy as possible to be handled by business experts. To do so, a CEP design layer should be added on top of this complexity leveraging the power of events combined with the knowledge of business experts, and making it easier to create event driven mashup applications.

To refine our requirements and select the best product for our final project solution we still have to define SocEDA use cases needs. Beside, we will keep watching mashup designer evolutions. Though, we know we may have whether to implement event introduction in the process design, or user-friendly layer above a more sophisticated design including events natively. In all cases, we know complex event integration will imply development.

In order to do so, we should go in the open source solutions direction and choose a product with a large community and preferably a strong documentation. Those aspects would help and make it easier to extend the product.

Another important aspect would be the target. If we want to make it easier in terms of CEP we also want to keep it easier in general. Basically, the product should be enough easy to be usable by developers as well as business users. We will need to keep the cursor between business users and developers, so between mashup tool and BPMN design tool. The best product should not be too complex to be only usable by developers, but it should be complex enough to build applications including CEP.

7. Relevant Standards

7.1. Mashup Standard (EMML)

The **Enterprise Mashup Markup Language** (EMML) is an XML markup language for creating enterprise mashups, which are software applications that consume and mash data from variety of sources, often performing logical or mathematical operations as well as presenting data. Mashed data produced by enterprise mashups are presented in graphical user interfaces as mashlets, widgets, or gadgets. EMML can also be considered a declarative mashup Domain Specific Language (DSL). A mashup DSL eliminates the need for complex, time-consuming, and repeatable procedural programming logic to create enterprise mashups. EMML also provides a declarative language for creating visual tools for enterprise mashups. EMML is an open language specification that is promoted by the Open Mashup Alliance with the eventual objective of submitting the specification to a recognized industry standards body. The language is free-to-use, including technologies that embed or use it.

High-level EMML language features include:

- Filter and sort data coming from heterogeneous services
- Join data across heterogeneous services and data formats
- Group and aggregate data using assorted functions
- Annotate original service data to enrich its semantic meaning
- Merge multiple data streams into consolidated datasets
- Split datasets to select individual data fields
- Embedded scripting support for JavaScript, JRuby, Groovy, XQuery
- Web clipping to scrape data from HTML pages
- Conditional statements - `If`/`Then`/`Else`, `While`, `ForEach`
- Parallel syntax for concurrent processing

EMML is primarily a XML-based declarative language, but also provides ability to encode complex logic using embedded scripting engines. XPath is the expression language used in EMML. The following example gives a flavor of EMML with *input*, *output* and direct invoke tags

EMML example:

```
<mashup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.openemml.org/2009-04-15/EMMLSchema ../schemas/EMMLSpec.xsd"
      xmlns="http://www.openemml.org/2009-04-15/EMMLSchema"
        xmlns:ns="http://webservices.amazon.com/AWSECommerceService/2007-07-16"
        name="AmazonWishlistSample">
  <!--This Mashup gets Amazon WishList for a given User-->

    <emml-meta name="author">Raj Krishnamurthy(raj@jackbe.com)</emml-meta>

    <input name="userId" type="string" default="rajmohan@mindspring.com" />
    <input name="accessId" type="string" default="1NWPHGDNQ6TXVWB94182"/>
    <output name="result" type="document"/>

    <!-- Invoke Amazon REST Service to get the given users wishlist
        Input variables  may be
            literals (e.g. 'AWSECommerceService', '2007-07-16', 'WishList' )
            variables (e.g. accessId, userId)
        Result is stored in 'result' variable
    -->
      <directinvoke endpoint="http://ecs.amazonaws.com/onca/xml" Service="AWSECommerceService"
                AWSAccessKeyId="$accessId"
                Operation="ListSearch"
                ListType="WishList"
                Email="$userId"
                ResponseGroup="ListInfo"
                outputvariable="result" />
    <!--
        Creates a custom result document containing only the desired elements.
    <constructor outputvariable="result">
            <ns:mylist>
                {$result/*:ListSearchResponse/*:Lists/*:List}
            </ns:mylist>
    </constructor>
    -->
</mashup>
```

Figure 29 : EMML example

7.2. Business Process Standard (BPMN/BPEL)

The primary goal of BPMN (**Business Process Modeling Notation**) is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

This specification represents the amalgamation of best practices within the business modeling community to define the notation and semantics of Collaboration diagrams, Process diagrams, and Choreography diagrams. The intent of BPMN is to standardize a business process modeling notation in the face of many different modeling notations and viewpoints. In doing so, BPMN will provide a simple means of communicating process information to other business users, process implementers, customers, and suppliers.

Another goal, but no less important, is to ensure that XML languages designed for the execution of business processes, such as WSBPEL (**Web Services Business Process Execution Language**), can be visualized with a business-oriented notation.

A Business Process Diagram can be made up of a set of (semi-) independent components, which are shown as separate Pools, each of which represents an orchestration Process. There is not a specific mapping of the diagram itself, but rather, each of these orchestration Processes maps to an individual WS-BPEL process (see Figure 30).

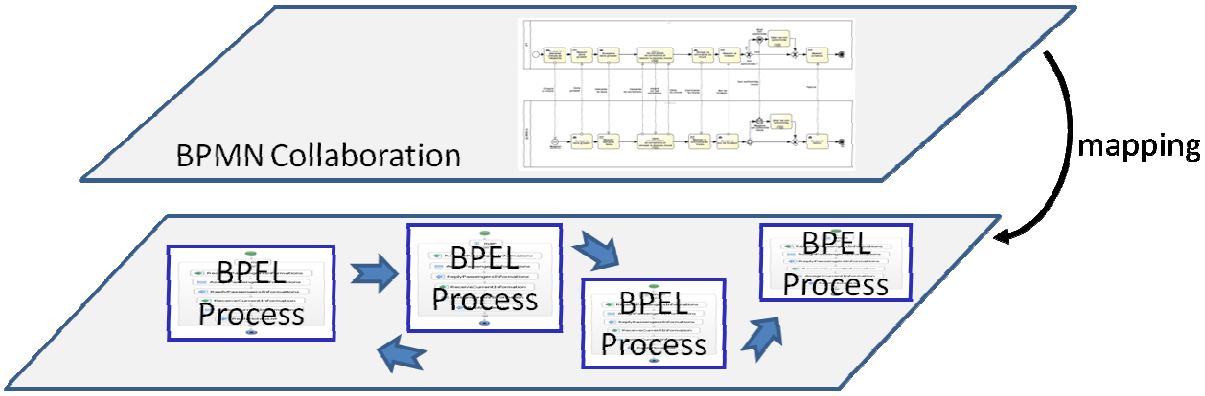

Figure 30 : BPMN to BPEL

8. References

8.1. Bibliography

[1] http://www.openajax.org/whitepapers/Ajax%20and%20Mashup%20Security.php

[2] Mashup definitions:

"*A mashup combines data or functionality from two or more external sources to create a new service.*" Information Management Magazine, Jan/Feb 2010,

"*Also referred to as composite applications. Custom applications that combine multiple, disparate data sources into something new and unique.*" Business Insights, The Future of Enterprise Mashups,

"Mashup is the remix or blending of existing information content to derive a single, integrated, new information content." The Art of Enterprise Information Architecture: A Systems-Based Approach for Unlocking Business Insight

"*Enterprise Mashups combine and remix data from databases, spreadsheets, websites, Web Services, RSS/Atom feeds, and unstructured sources that deliver actionable information for better decision-making*." Open Mashup Alliance,

"A mashup is a (…) user-facing and user-directed process, where an end user directs an application to combine and aggregate data from various sources (RSS feeds, Atom feeds, web pages, etc.) and combines the data with a presentation to provide a comprehensive view of the aggregation." Applied SOA: Service-Oriented Architecture and Design Strategies

[3] Information Security Management Handbook, Sixth Edition, Volume 3 - by Harold F. Tipton and Micki Krause (eds) - Auerbach Publications © 2009

[4] Reshaping Your Business With Web 2.0: Using the New Collaborative Technologies to Lead Business Transformation - by Vince Casarez, Billy Cripe, Billy Cripe and Philipp Weckerle - McGraw-Hill/Osborne © 2009:
"Process mashups Often called SOA orchestration, mashup assemblers (developers or technical business users) leverage a set of existing Business Process Execution Language (BPEL) processes and connect them together in the context of their required task."

[5] Foundations of Popfly: Rapid Mashup Development - Eric Griffin, Apress© 2008

[6] Generic Business Model Types for Enterprise Mashup Intermediaries - Volker Hoyer and Katarina Stanoevska-Slabeva - Value Creation in E-Business Management, 15th Americas Conference on Information Systems, AMCIS 2009, SIGeBIZ track, San Francisco, CA, USA

[7] A Rule-Based Approach of Creating and Executing Mashups - Emilian Pascalau and Adrian Giurca - IFIP Advances in Information and Communication Technology, 2009, Volume 305, Software Services for e-Business and e-Society

[8] The Future of Enterprise Mashups - Business Insights - 21 Aug 2009- E-Strategies for Resource Management Systems: Planning and Implementation - by Eshaa M. Alkhalifa (ed) - IGI Global © 2011

[9] Quality-Based Recommendations for Mashup Composition - Matteo Picozzi, Marta Rodolfi, Cinzia Cappiello, and Maristella Matera - Lecture Notes in Computer Science, 2010
[10] DreamFace : http://dreamface-interactive.com/
[11] Presto : http://www.jackbe.com/enterprise-mashup/
[12] WSO2 mashup solution : http://wso2.com/products/
[13] IBM WebSphere Smash: http://www-01.ibm.com/software/webservers/smash/
[14] Cordys MashApps Composer: http://www.cordysprocessfactory.com/cloud-applications
[15] Bonita Soft: http://www.bonitasoft.com/
[16] GEasyBPMN Editor : http://www.petalslink.com/en
[17] Open Mashup Alliance : http://www.openmashup.org/
[18] Convertigo: http://www.convertigo.com/
[19] Kapow: http://kapowsoftware.com/

8.2. Illustrations table

9. Appendix

9.1. Mashup tools base list

Name	Description	URL	Stopped
Apatar	Open source lightweight software designed for business users and programmers to mash data from various data sources and format	http://www.apatar.com/product.html	
Boomi Atomsphere	Online platform allowing graphical workflow composition (EAI in SaaS mode)	http://www.boomi.com/products	
Bungee Connect	Developer oriented Eclipse-based environment for building web applications by mashing Web Services and data sources	http://www.bungeeconnect.com/index.html	
Convertigo Enterprise Mashup studio and composer	A eclipse-based environment to create widgets and a composer to build applications by making widgets interacting	http://www.convertigo.com/en/crm/mashup-scm.html	
Cordys MashApps Composer	Allows the creation of composite applications through forms-driven business processes design	http://www.cordysprocessfactory.com/cloud-applications	
Denodo Entreprise data mashup	A platform focused new business services creation by integrating existing data	http://www.denodo.com/	
DreamFace	Widget-based framework to build, use, and distribute enterprise Web 2.0 applications and mashups	http://dreamface-interactive.com/	
Encanvas Mashup Software	Web user interface allowing data mashing	http://www.encanvas.com/main/products/mashups.html	
Extensio	A platform for data extraction, integration, and delivery	http://www.extensio.com/	
IBM sMash	Web environment allowing sequencing of Web components as feeds, REST services. An open source version of sMash is ProjectZero	http://www-01.ibm.com/software/webservers/smash/	
Intel Mash Maker	An experimental research project for enabling the easy creation of mashups ("mashups for the masses")	http://software.intel.com/en-us/articles/intel-mash-maker-mashups-for-the-masses/	X
JackBe Presto	Software to let users create, consume and customize enterprise mashups. Presto is data plus presentation mashing oriented, compositions can be realized graphically	http://www.jackbe.com/enterprise-mashup/	

jMaki	Developers oriented Ajax framework to build Web 2.0 applications by mashing presentation components.	http://ajax.java.net/	
Just System XFY	XML based architecture framework for creating dynamic composite documents	http://na.justsystems.com/content-xfy	
Marmite	A research prototype of a mashup creation tool for non programmers. Marmite is data plus presentation mashing oriented	http://www.cs.cmu.edu/~jasonh/projects/marmite/	X
Microsoft Popfly	A browser mashup tool whose drag-and-drop widgets have some similarity to Yahoo! Pipes but that puts more of an emphasis on presentation gadgets	http://www.xaml.fr/popfly.html	X
Nexaweb Enterprise Web Suite	An open web development platform allowing Composite and Enterprise Mashup Application Development	http://www.nexaweb.com/products/enterprise-web-suite/	
Openkapow	Desktop software for data integration using bots	http://kapowsoftware.com/	
Proto	Desktop data mashups, for data management, analysis and reporting tasks	http://www.protosw.com/	
RSSBus	Tools to "generate, manage, orchestrate, and pipeline RSS feeds"	http://rssbus.com/	
Serena Mashup Composer	Software for creating business mashups based on web services orchestration	http://www2.serena.com/geo/fr/products/mashup-composer/features/feature-integrated-server.html	?
WaveMaker	Allows creating web applications based on a drag and drop interface, using widgets and templates. Allows web services call, database access and simple orchestration but in more complex manner	http://www.wavemaker.com/	
WSO2 Mashup Server	An open source platform for creating and deploying "web services mashups" and delivering enterprise-class service composition	http://wso2.com/products/mashup-server/	
Yahoo Pipes	A composition tool to graphically aggregate, manipulate, and mashup Web content	http://pipes.yahoo.com/pipes/	

Based on Orange researched plus sources including “Pro Web 2.0 Mashups: Remixing Data and Web Services” - By Raymond Yee, and “Top 10 Enterprise Mashup software products” NDMC white paper.